175 ptas.
mi computer
54
CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR
FIRE
Cheetah
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen V - Fascículo 54

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-007-4 (tomo 5)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 238501
Impreso en España - Printed in Spain - Enero 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Gran final

Por último analizaremos algunos de los sistemas más avanzados con tecnología digital incorporada

De los recientes desarrollos en la música electrónica, el más significativo se ha producido en el área de la grabación digital. No sólo se ha mejorado enormemente la calidad de grabación del sonido, sino que el significado de la palabra "grabación" ha cambiado para abarcar diversidad de técnicas. Si entendemos que grabación significa "codificación digital del sonido y su organización en forma de música", podemos empezar a captar lo que está sucediendo con la música en la década de los ochenta.

Desde la segunda guerra mundial, la grabación de sonido en cinta magnética ha sido la norma, con formatos que van desde la más diminuta microcassette hasta los grandes carretes de cinta de 24 pistas utilizados en los estudios de grabación profesionales. Cuando se realiza una grabación en cinta magnética, las pequeñas partículas de óxido de metal de la superficie de la cinta se disponen en complejos patrones que son análogos a las formas de onda de sonido que representan. A medida que la cinta pasa por el cabezal de reproducción de un dispositivo de grabación, estos patrones se convierten en series de voltajes eléctricos. Estos voltajes entran posteriormente a los altavoces, los cuales reproducen el sonido grabado en la cinta.

Dado que la disposición de las partículas se puede conocer con bastante precisión en relación al cabezal de reproducción de cinta, es bastante sencillo descubrir dónde se encuentra un determinado sonido en un trozo de cinta magnética, de modo que el empalmado y la edición de cinta con hojas de afeitar desimantadas se ha convertido en una importante habilidad que los ingenieros de sonido deben adquirir.

En grabación digital, el sonido se codifica numéricamente a lo largo de la cinta, y el cabezal de reproducción se convierte en un convertidor de digital a analógico. La entrada a los altavoces es la misma que antes, pero éstos utilizan voltajes generados por el convertidor D/A. Siempre y cuando haya datos suficientes para convertir, la cinta digital logra una reproducción enormemente superior a la que se consigue con cinta magnética. Y, puesto que los datos permanecen intactos, la cinta se puede copiar digitalmente cientos de veces sin pérdida de calidad. Con la cinta magnética, sin embargo, cada "generación" de copias agrega "ruidos" y distorsión a la grabación.

Este tipo de degradación por ruido ha constituido un problema acuciante para los ingenieros de sonido durante muchos años. Ahora que se ha resuelto, muchos de los mejores estudios de graba-



Synclavier
El Synclavier, de New England Digital, se considera como uno de los más avanzados del mundo. Aparte de las funciones usuales de sintetizador, la máquina posee la capacidad de almacenar hasta 10 Megabytes de sonido en disco

Fairlight CMI
El Fairlight CMI fue uno de los primeros sistemas de música por ordenador. Su sistema operativo es conducido por menú, permitiendo diversas opciones, desde control por teclado a dibujo de formas de onda. Asimismo, la máquina tiene la facilidad de producir salidas impresas



Yamaha KX5
La interface MIDI del Yamaha KX5 proporciona un vínculo entre sintetizadores y el Yamaha CX5. Este dispositivo también se utilizará para conectar en interface el ordenador personal Yamaha MSX cuando éste se comercialice en Europa

Roland MSQ-700
El Roland MSQ-700 se presenta como el primer secuenciador del mundo compatible con la MIDI. El MSQ-700 posee una gama completa de facilidades MIDI y puede almacenar hasta 6 500 notas





Drumulator
La máquina de ritmos Drumulator, de Emu Systems, tiene una capacidad de memoria de 10 088 notas en 64 canciones. También hay facilidades para permitir la inserción de ROM adicionales que proporcionen efectos especiales, como percusión latina o africana

Ian McKinnell

ción han instalado grabadoras de cinta digitales de 24 pistas. Utilizándolas, la reproducción del sonido es tan exacta que a los ingenieros les resulta imposible distinguir si un sonido proveniente de los altavoces monitores del estudio lo está produciendo un músico en el área de grabación o si son fruto de la reproducción de una cinta digital. Pero han aparecido nuevos problemas: ya no se puede "ver" dónde estan situados los sonidos en una cinta digital, lo que hace que la edición resulte más difícil; por otra parte el empalmado se está convirtiendo en una habilidad obsoleta. Otra dificultad es el "ruido del estudio", una característica indeseable y a menudo también inaudible de algunos equipos de audio. La cinta magnética no era lo suficientemente sensible como para detectarlo, pero las grabaciones digitales sí tienden a captarlo.

Mientras que la grabación en 24 pistas sigue siendo todavía prerrogativa de los estudios caros, la grabación digital en una única pista, de la misma calidad, está a disposición de cualquier usuario de un video Betamax. La cinta de video es un soporte digital y, como tal, se puede utilizar para codificar cualquier tipo de datos. El Sony PCM (*Pulse Code Modulator*: modulador de códigos de impulsos) es una unidad que convierte a un video Betamax en una grabadora de cinta de audio. Esta unidad posee el potencial para hacer que las grabadoras analógicas de tamaño similar queden obsoletas.

La codificación digital del sonido, o muestreo (*sampling*), es el corazón del Fairlight CMI (Computer Musical Instrument), uno de los sistemas avanzados más conocidos. El Fairlight puede muestrear cualquier sonido de una duración de hasta dos segundos y reproducirlo luego a través de una escala de altura de seis octavas. El muestreo representa un avance decisivo en el campo de la música electrónica. Durante años, los ingenieros y los músicos han estado tratando de simular el sonido de instrumentos de cuerda o de viento y de madera utilizando sintetizadores, y en algunos casos han estado muy cerca de lograr su objetivo. Pero el muestreo no sólo proporciona una notable reproducción del sonido de cuerda, sino que puede producir el sonido de un violín en especial. Además, en algunos casos, puede reproducir el sonido de un determinado músico en una determinada habitación. En el primer capítulo de esta serie vimos cómo los compositores de *musique concrète* de los años cincuenta se pasaban semanas empalmando diminutos retazos de cinta grabada, produciendo finalmente obras a gran escala. Ahora la manipulación de muestras por ordenador permitirá que un compositor produzca resultados similares en cuestión de minutos.

Ecos del pasado
Este fragmento está extraído de *The new Atlantis* (La nueva Atlántida), una visión utópica escrita por el filósofo inglés Francis Bacon (1561-1626). Sus descripciones de los sonidos parecen anticipar el extraordinario poder y versatilidad de la música electrónica de hoy


Biblioteca Mary Evans

De LA NUEVA ATLÁNTIDA, de Francis Bacon, publicado en 1624:

También tenemos Casas de Sonido, donde practicamos y demostramos todos los sonidos, y su generación. Tenemos armonías, que ustedes no poseen, de cuartos de sonido, y de porciones de sonidos aún inferiores. Diversos instrumentos de música que ustedes desconocen igualmente, algunos de ellos más dulces que cualquiera que ustedes posean; junto con campanas y carillones que son dulces y exquisitos. Representamos pequeños sonidos como grandes y profundos, y los sonidos grandes debilitados y finos. Hacemos diversos temblores y sonidos de gorjeos, que en su original son enteros. Representamos e imitamos todos los sonidos articulados y las letras y las voces y las notas de bestias y pájaros. Contamos con ciertas ayudas, que colocadas junto a la oreja engrandecen cuanto se escucha. Asimismo tenemos diversos ecos extraños y artificiales, que reflejan la voz muchas veces y como si la estuvieran lanzando al aire. Y algunos que devuelven la voz más fuerte cuando llega, algunos más aguda y otros más profunda. Y algunos que dan la voz con letras o sonido articulado que difieren de la que reciben; también tenemos medios para transmitir sonidos por conductos y tubos, en extrañas líneas y distancias.

Instrumentos de muestreo

Un instrumento de muestreo como el Fairlight puede superar las limitaciones naturales de los instrumentos musicales. Por ejemplo, para un flautista es bastante fácil producir una calidad de tono cálida en el extremo inferior de la escala de la flauta. Sin embargo, para un músico, independientemente de lo hábil que sea, es imposible obtener este tipo de sonido dos octavas más arriba en el extremo superior de la escala del instrumento: el diseño físico de la flauta lo impide. Un usuario del Fairlight, por otra parte, puede muestrear el tono bajo y después transportarlo hacia arriba dos octavas en el teclado. El resultado seguirá sonando como una flauta, pero una clase de flauta que no puede existir en el mundo analógico "real".

El Fairlight puede suministrar una visualización en pantalla de cualquiera de los sonidos muestreados, que se almacenan en discos de 8". Las diferentes características de un sonido individual pueden ser examinadas en sucesión: con frecuencia resulta más fácil decir qué es lo que está "mal" en un sonido determinado mirándolo en vez de escuchándolo. Muchos sonidos es necesario que sean más largos que su duración de muestra original de dos segundos. Viendo cómo están relacionadas las diferentes formas de onda dentro del sonido, se puede seleccionar un punto en el cual hacer que el sonido empiece a "buclar" (*looping*) o a repetirse. De seleccionarse el punto correcto, éste dará la ilusión de una auténtica continuidad. El sonido analógico no se puede comportar así, por supuesto, de modo que el "buclado" puede darle a la música que esté produciendo una dimensión insólita.

El usuario del Fairlight tiene dos formas de entrar la música, además de tocarla en "tiempo real" en el teclado. El primer método, conocido como *Page R* (página R), ofrece la visualización de un pentagrama, y el usuario entra las notas en éste desde el teclado de música. Cualquier error de tiempos lo "ordena" automáticamente el ordenador de acuerdo a la métrica o compás que el usuario haya especificado.



El segundo método consiste en utilizar un MCL (*Music Composition Language*: lenguaje para composición musical). El Fairlight MCL exige que cada evento de nota se entre mediante el teclado alfanumérico, pero permite modificar el tiempo y la acentuación de nota a nota. El Fairlight tiene una capacidad de ocho voces, de modo que un usuario podría entrar ocho frases distintas, tocadas por ocho sonidos o "instrumentos" distintos. Cada uno de éstos puede estar ligeramente fuera de tiempo (por apenas unos milisegundos) respecto a los otros, y toda la ejecución es coordinada por el reloj interno del Fairlight.

Quizá usted pregunte qué sentido tiene tocar música de esta forma "incorrecta", sobre todo cuando el ejecutante es un ordenador. La respuesta a esta cuestión es que las personas jamás tocaron exactamente al compás, y una de las características que definen la ejecución (especialmente entre los músicos de jazz y algunos clásicos) es la manera en la cual un músico puede modificar el tiempo musical cuando ejecuta un pasaje. Un sistema operativo como el Fairlight proporciona una forma en la cual se pueden simular ciertos estilos de ejecución. Estas simulaciones se pueden emplear en trabajos experimentales y de investigación, así como se utilizan las simulaciones por ordenador en el diseño de carrocerías de automóviles, de alas de aviones y de los escudos contra las altas temperaturas de las "lanzaderas espaciales" *(space shuttles)*.

Muchos músicos están justamente preocupados por la eventualidad de que instrumentos como el Fairlight lleguen a reemplazar a las personas, especialmente a medida que se vaya desarrollando la capacidad de simulación de tales equipos. Grupos como Wang Chung, Duran Duran y Culture Club utilizan Fairlight como parte de su proceso de producción, y con frecuencia resulta imposible decir qué es lo que realmente se está tocando y qué es lo que está ejecutando el Fairlight. No obstante, una vez que un usuario conoce el sistema operativo, se hace evidente que el Fairlight es mucho más que un mero instrumento musical nuevo y que tiene un potencial en gran medida todavía inexplorado.

Si bien es el más conocido, el Fairlight no es el único instrumento de estas características que existe. El sistema Synclavier —que cuesta casi el doble que el Fairlight— posee facilidades similares, pero a mayor escala. Los datos se almacenan utilizando discos rígidos Winchester con una capacidad de 40 megabytes. Con el propio sistema Synclavier se podría producir y grabar un álbum completo, haciendo totalmente innecesaria una avanzada máquina digital de 24 pistas.

Pero hasta el Synclavier tiene sus limitaciones. En la actualidad todos los instrumentos de muestreo existentes poseen un rasgo común: reproducen el sonido muestreado de la forma más cercana posible al original, a menos que el usuario haya intervenido para realizar una modificación específica. Pero un trompetista, en medio de una actuación en vivo, puede introducir una considerable diferencia en el siguiente sonido a tocar simplemente cambiando el control de la respiración o la posición de los labios. Un trompetista competente hace esto prácticamente sin pensarlo. El siguiente paso para los instrumentos de muestreo sería, por consiguiente, producir un sistema "que responda al músico".

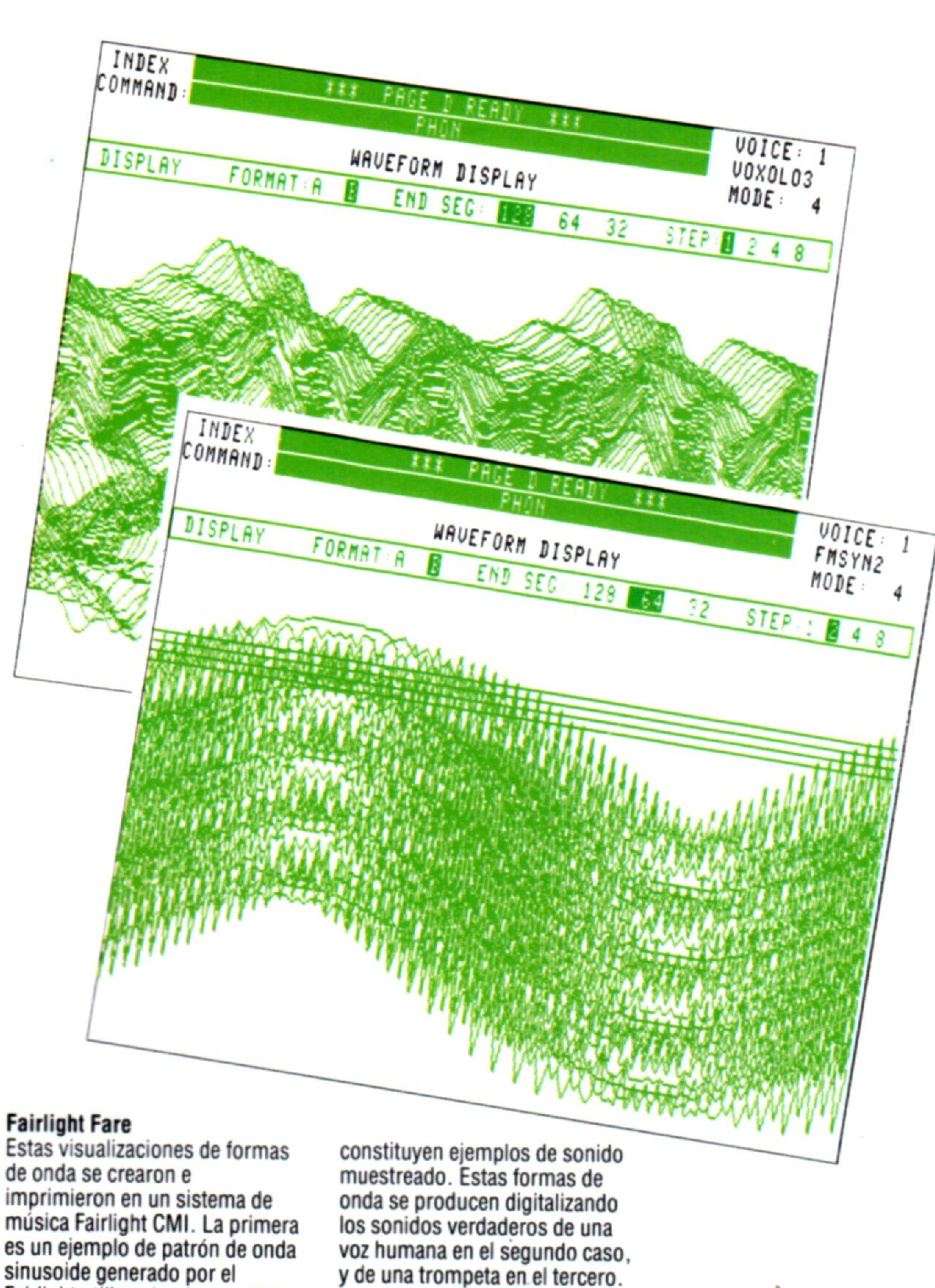
El Kurzweil, que aún es un sistema modélico, incorpora un programa de reconocimiento de patrones. Ello significa que cuando se toca una nota en el teclado de música, se exploran una cantidad de muestras diferentes y se combinan las características de cada muestra para producir el sonido individual. El tipo de características seleccionadas reflejará la forma en que se toque la música. Tal sistema se asemejaría cada vez más al carácter y la sensación de un auténtico instrumento acústico. La única diferencia es que ningún piano vertical ni piano de cola es exactamente igual a otro. Todos los "pianos" Kurzweil CMI serían idénticos en carácter y sensación, a menos que los usuarios desarrollen un método para escribir un software de su "propio carácter y sensación".

Se rumorea que Lucasfilm, productora de las películas *La guerra de las galaxias* e *Indiana Jones y el templo maldito*, está desarrollando un sistema aún más avanzado que el Fairlight, Synclavier y Kurzweil reunidos. Se espera que este sistema, denominado ASP (*Audio Signal Processor*: procesador de señales de audio), incorpore, en un solo sistema operativo complejo, todos los tipos de facilidades para sonido digital musical de que disponen actualmente sólo los grandes estudios musicales. De modo que, así como los ordenadores y los sintetizadores de los años cincuenta ocupaban el espacio equivalente a la superficie de varias habitaciones, y ahora ocupan sólo la superficie de un escritorio, podemos esperar que el estudio de grabación del futuro sea un paquete portátil.

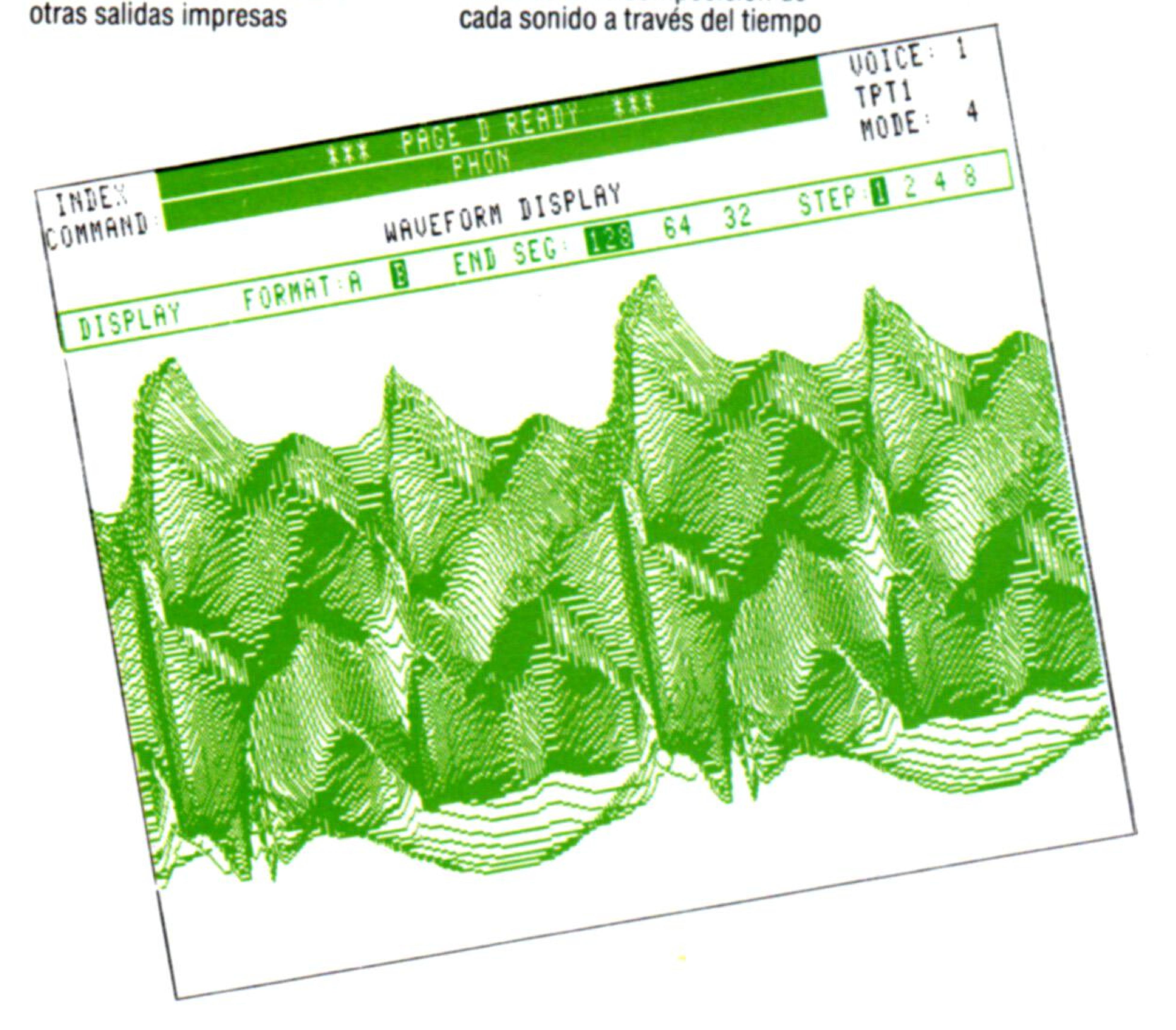



```
'B=24:G=1/24:V=15
R,36:<(Bf4:D4),5:(Bf4:D4),7
R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(F4:C4:Ef4)V13,7/3
(Bf4:D4),5:(Bf4:D4),7
R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(G4:B4:Ef4)V13,7/3
(D4:F5),5:(D4:F5),7
R,5:(D4:F5)V10,7/3:(D4:F5)V11,7/3:(F4:C4)V13,7/3>2
(Bf4:D4),5:(Bf4:D4),7
R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(F4:C4),7/3:T=+3>3:T=0
<(Bf4:D4),7/3:(Bf4:D4),7
<(Bf4:D4:A5),5:(C4:Ef4),7/2:(D4:Bf5),7/2>2
(Bf4:D4:A5),5:(Bf4:D4),7
(C4:Ef4:Bf5),5:(Bf4:D4),7/3:(Ef4:C4),7/3:(A5:C5),7/3
(C4:Ef4:Bf5),5:(Bf4:D4),7
(C4:Ef4:Bf5),5:(Bf4:D4),7/4:R,5/6
(Bf4:D4),7/8:R,5/12
(G4:B4:Ef4),75/24:V=15
<(Bf4:D4),5:(Bf4:D4),7
R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(F4:C4)V13,7/3
(Bf4:D4),5:(Bf4:D4),7
R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(F4:C4:Ef4)V13,7/3
(Bf4:D4),5:(Bf4:D4),7
R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(G4:B4:Ef4)V13,7/3
(D4:F5),5:(D4:F5),7
R,5:(D4:F5)V10,7/3:(D4:F5)V11,7/3:(F4:C4)V13,7/3>2
(Bf4:D4),5:(Bf4:D4),7
R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3
```

Fairlight MCL
El lenguaje para composición musical utilizado por el Fairlight CMI es conducido por menú, de modo que las opciones se eligen a partir de menús claros y descriptivos que aparecen en la pantalla. Las formas de onda visualizadas en la página 1064 se generaron con una lista de parámetros de sonido en sentencias DATA, similar a un listado en BASIC, visualizándose e imprimiéndose luego utilizando instrucciones tomadas del menú



Fairlight Fare
Estas visualizaciones de formas de onda se crearon e imprimieron en un sistema de música Fairlight CMI. La primera es un ejemplo de patrón de onda sinusoide generado por el Fairlight utilizando síntesis FM (p. 1041). Este sonido se fabricó mezclando formas de onda electrónicamente. Las otras salidas impresas constituyen ejemplos de sonido muestreado. Estas formas de onda se producen digitalizando los sonidos verdaderos de una voz humana en el segundo caso, y de una trompeta en el tercero. Las visualizaciones son "tridimensionales" o topográficas, representando los cambios en la composición de cada sonido a través del tiempo



La incorporación de la tecnología digital no se ha limitado exclusivamente a los dispositivos y sistemas para generación de sonido. Los modernos estudios de grabación generalmente incluyen un cierto número de unidades para tratamiento de sonido como parte de su equipo básico. Un ejemplo es la *unidad de reverberación*. La música se encamina a través de esta unidad para agregar eco o "reverberación", y los guitarristas *rockabilly* y los productores de *dub reggae* han dependido de este tratamiento para darle a su música ese particular sonido.

El Quantec es una unidad digital que, en vez de meramente proporcionar reverberación, simula los tipos de reverberación que ocurren en habitaciones y espacios de diferentes dimensiones. Su "espacio acústico" más pequeño es una caja con un volumen de un metro cúbico, y las simulaciones preestablecidas incluyen dimensiones típicas de salas de estar, auditorios, hangares de aviones y catedrales. La característica más interesante del Quantec es la facilidad que ofrece para prolongar la reverberación bastante más allá de los límites acústicos o físicos naturales. De este modo, si se produce un sonido en la simulación de las dimensiones de catedral, y si se maximiza el tiempo de reverberación, todo el evento durará varios minutos: el efecto es algo así como escuchar un eco creado ¡por diez Grandes Cañones del Colorado unidos!

Se dice que en el Medio Oeste norteamericano de los años cincuenta había un pequeño estudio de *rockabilly* que se había construido cerca de un silo de cereales. Era este inmenso espacio el que le proporcionaba al estudio su clásico sonido *rockabilly* lleno de ecos. El empleo moderno de unidades digitales como el Quantec es más trivial, ciñéndose principalmente a las etapas de post-producción de los trabajos de cine y televisión. Se puede filmar a los actores hablando en el entorno acústicamente "muerto" de un estudio, y se puede tratar la banda de sonido después para proporcionar una resonancia acústica que se adapte al espacio en el cual se supone que transcurre la acción.

Valores "humanos"

Muchas personas, músicos y no músicos indistintamente, piensan que la tecnología digital aplicada a la música tiene un efecto pernicioso. Aducen que esta música está creada y tocada de una manera tan artificial que los auténticos valores "humanos" de la espontaneidad y la expresividad quedan sepultados en la carrera por adquirir niveles de control técnico cada vez mayores. Éste es un argumento convincente, pero vale la pena considerar un medio visual como el cine para juzgar si es éste realmente el caso.

Durante varias décadas la tecnología cinematográfica les ha permitido a los directores rodar escenas desde cualquier ángulo, acercar la cámara para filmar con detalle y girarla para captar grandes panorámicas. Ha sido posible repetir secuencias, ralentizarlas, acelerarlas, hacerlas retroceder y montar la película de modo que en fracciones de segundo puedan ocurrir las yuxtaposiciones visuales más insólitas. Sólo recientemente, con la tecnología digital, los compositores de música han comenzado a ejercer sobre el sonido un control comparable, de modo que es posible que la música digital finalmente se considere bajo el mismo prisma.



Control de motores

Ahora veremos qué software se requiere para conmutar motores y desarrollaremos un sistema sencillo de control de realimentación

El control por microprocesador de dispositivos externos actualmente es común en la industria, con aplicaciones que van desde contar las botellas que pasan por una cinta transportadora hasta soldar carrocerías de automóviles. Los principios de todos los sistemas de control son, en primer lugar, entrar datos desde el mundo exterior en un formato comprensible para un sistema de microprocesador; en segundo término analizar estos datos, y, finalmente, instigar acciones externas basadas en ese análisis. Si estas tres actividades se repiten en un ciclo continuo, entonces tenemos un sistema que se conoce como *control de realimentación* o de *feedback*.

Para ilustrar el principio del control de realimentación tomemos el ejemplo de calentar una cacerola con sopa en una cocina. Con el fin de cocer la sopa, el calor suministrado debe ser suficiente para hacerla hervir, pero no tanto como para que se derrame por los bordes de la cacerola. Si fuéramos a llevar a cabo esta tarea nosotros mismos, inicialmente aplicaríamos calor máximo a la cacerola hasta que la sopa comenzara a hervir y entonces lo reduciríamos hasta dejar el líquido hirviendo suavemente. Si en cualquier momento la sopa comenzara a hervir demasiado, reduciríamos aún más el calor. Al hacer esta clase de operación nosotros controlamos visualmente el estado de la sopa, analizamos lo que vemos y emprendemos la acción apropiada. Repetiríamos estas acciones hasta que el alimento estuviera listo para ser ingerido. Un microordenador controlaría la cocción de la sopa de una forma similar, aunque no idéntica. La principal diferencia radicaría en la forma de controlar el estado de la sopa. Mientras que nosotros podemos mirarla y formular un juicio valiéndonos de nuestra experiencia, un sistema informático tendría que utilizar otro método basado en propiedades físicas fácilmente determinables, tales como la temperatura. Antes de que se informatizara el control de la sopa, alguien debería llevar a cabo experimentos iniciales para determinar la temperatura que le corresponde a una suave ebullición constante y a partir de qué temperatura la sopa comenzará a hervir peligrosamente. A partir de entonces, sin embargo, el ordenador asumirá la tarea, siempre y cuando estén disponibles los dispositivos apropiados que permitan medir la temperatura y regular el calor.

Se pueden controlar motores de varias formas utilizando la caja buffer (véase p. 1026) y la caja de salida de bajo voltaje (véase p. 1054) que hemos construido. Un motor de juego de tren eléctrico o Lego es ideal para comprobar el software que diseñaremos. Sólo necesitamos asegurar que el motor que conectemos a la caja de salida tenga un voltaje nominal igual o mayor que el voltaje de entrada del transformador. Conectando los dos terminales del motor a la línea 0 de la caja de salida, podemos encender y apagar el motor utilizando las teclas "Z" y "X" del teclado.

BBC MICRO

```
10 REM MOTOR SENCILLO BBC
20 RDD=&FE62:REGDAT=&FE60
30 ?RDD=255:REM TODAS LAS LINEAS SALIDA
40 ?REGDAT=0:REM MOTOR APAGADO
50 REPEAT
60 A$=INKEY$(1):REM PULSACION DE TECLA?
70 IF A$="Z"THEN ?REGDAT=1: REM ENCENDER
80 UNTIL A$="X"
90 ?REGDAT=0:REM APAGAR
```

COMMODORE 64

```
10 REM MOTOR SENCILLO CBM64
20 RDD=56579:REGDAT=56577
30 POKERDD,255:REM TODAS LAS LINEAS SALIDA
40 POKEREGDAT,0:REM MOTOR APAGADO
50 GETA$:IFA$ <> "Z" ANDA$ <> "X" THEN
   50:REM ESPERAR PULSACION TECLA
60 IF A$="Z" THEN POKEREGDAT,1:GOTO50
70 IF A$="X" THEN POKEREGDAT,0:END
```

Ejercicios

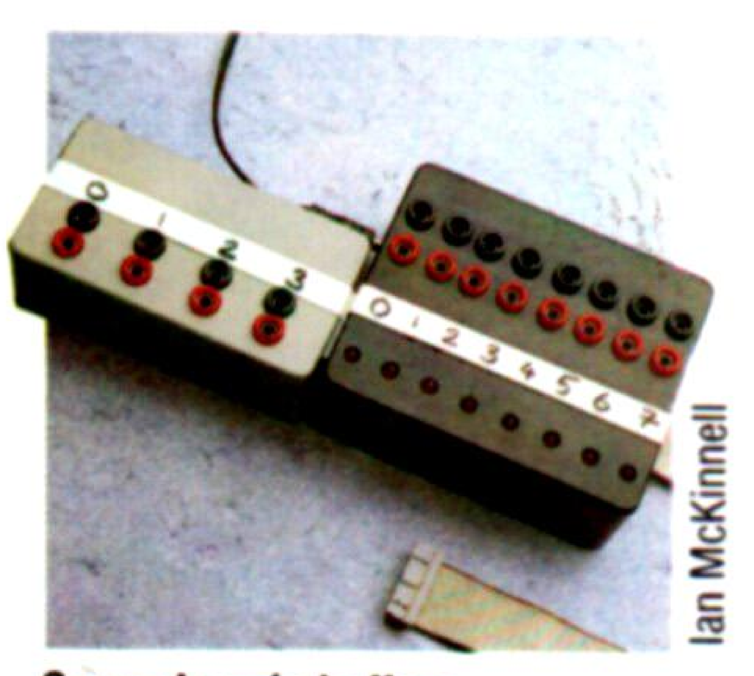
Ahora estamos en condiciones de diseñar algunos elegantes experimentos de control que se basan en entrada y salida utilizando la caja buffer y la caja de salida de bajo voltaje. Los sensores a los que hacen referencia pueden ser sensores de presión o interruptores "de lengüeta" operados por un pequeño imán. Tales dispositivos se pueden conseguir en tiendas de electricidad y electrónica a un precio muy reducido. Los interruptores sensibles al calor operan haciendo un contacto interno entre los terminales cuando se alcanza una cierta temperatura.

1) Escriba un programa para hacer que un vehículo corra hacia atrás y hacia adelante entre dos sensores colocados en su camino.

2) Escriba un programa para hacer que un vehículo se detenga justo encima de un sensor, haciendo marcha atrás de ser necesario.

3) Escriba un programa para mantener una cubeta de agua entre 70 y 100 °C usando un calentador de bajo voltaje y dos interruptores sensibles al calor.

4) Escriba un programa para calcular la velocidad en metros por segundo de un vehículo que viaje entre dos puntos. (Necesitará conocer la distancia y registrar el tiempo invertido en recorrerla.)



Ian McKinnell

Sacando más buffers
La caja buffer, nuestra primera construcción, está destinada a proteger la circuitería del ordenador contra corrientes de entrada o salida excesivas. Su fuente de alimentación eléctrica independiente activa la caja de salida (el proyecto más reciente) y permite la conmutación controlada por software de la salida de 12 voltios

Podemos controlar la dirección del motor conectando sus terminales a líneas adyacentes de la caja de salida. El diagrama ilustra estas conexiones. Y utilizaremos un interruptor simple conectado a la línea 7 de la caja buffer para controlar la dirección.

En el siguiente programa, un valor de 1 en el registro de datos hace que fluya corriente en un sentido a través del motor. Colocando un valor de 2 en el registro de datos, la corriente fluirá en el sentido contrario. El programa investiga repetidamente la línea 7 y sólo coloca un 2 en el registro de datos cuando la línea está baja (es decir, el interruptor está cerrado). De esta forma, el cierre y la apertura del interruptor controlan la dirección del motor. Éste es un ejemplo muy sencillo de un sistema de control de realimentación.

BBC MICRO

```
10 REM MOTORES DIRIGIDOS BBC
20 RDD=&FE62:REGDAT=&FE60
30 ?RDD=127:REM LINEA 7 ENTRADA
40 ?REGDAT=0:REM APAGAR
50 A$=GET$:REM ESPERAR PULSACION TECLA
60 REPEAT
70 A$=INKEY$(1)
80 IF(?REGDAT AND 128)=0 THEN DIR=
   2 ELSE DIR=1
90 ?REGDAT=DIR
100 UNTIL A$="X":REM PULSAR X PARA ACABAR
110 ?REGDAT=0:REM APAGAR
```

COMMODORE 64

```
10 REM MOTORES DIRIGIDOS CBM64
20 RDD=56579:REGDAT=56577
30 POKERDD,127:REM LINEA 7 ENTRADA
40 POKEREGDAT,0:REM TODO APAGADO
50 GETA$:IFA$="" THEN50:REM ESPERAR
   PULSACION TECLA
60 GETA$
70 IF(PEEK(REGDAT)AND128)=0THENPOKEREGDAT,
   2:GOTO90
80 POKEREGDAT,1
90 IFA$ <> "X"THEN60
100 POKEREGDAT,0:REM APAGAR
```

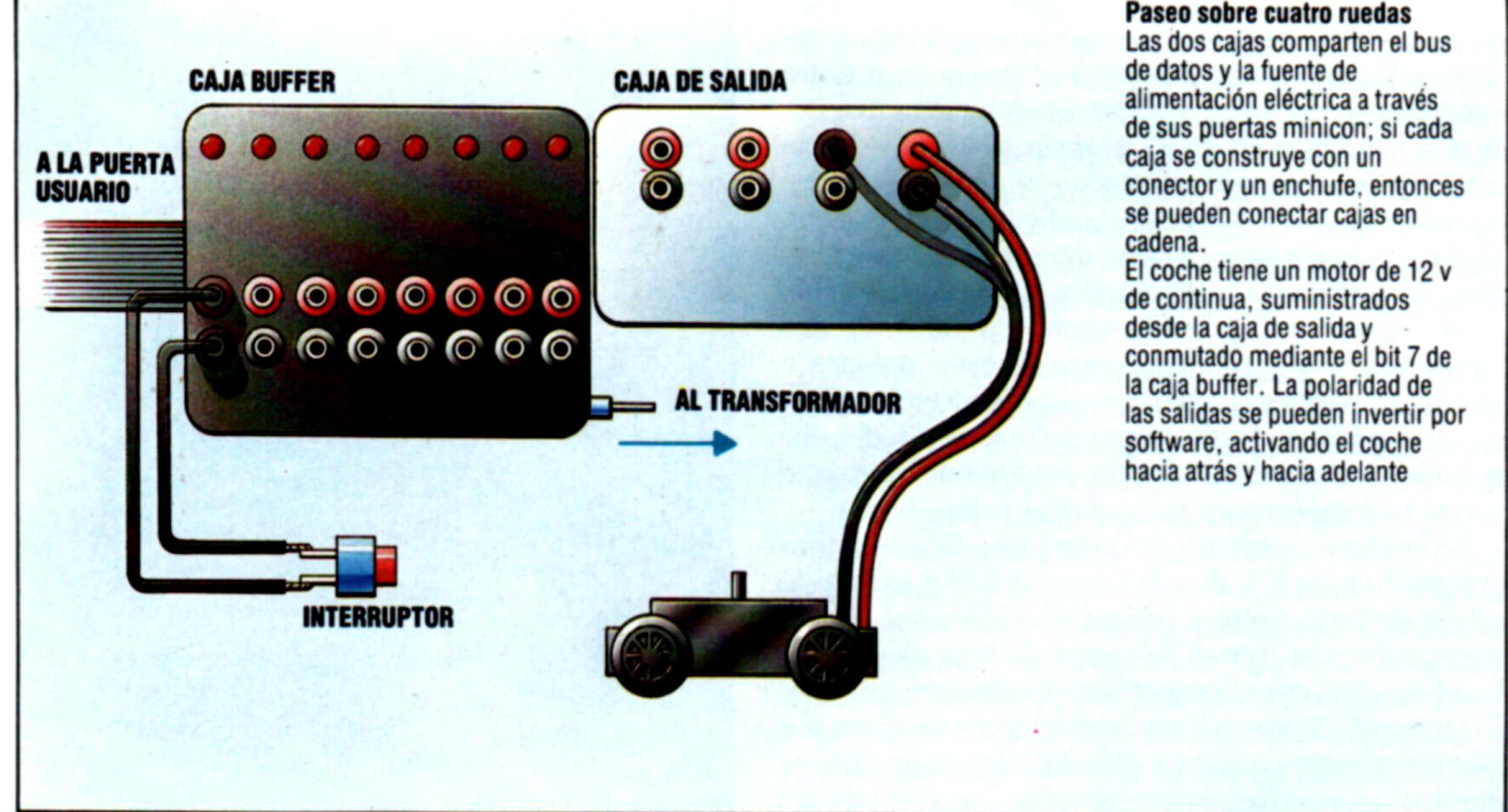
Además de controlar la dirección de un motor, también podemos controlar su velocidad directamente desde la caja de salida. Ello no exige dispositivos complicados, como convertidores de digital a analógico para controlar el suministro de corriente a los motores. En cambio, podemos enviarle impulsos al motor, encendiéndolo y apagándolo en rápida sucesión. Si hacemos esto con la suficiente rapidez, parecerá que éste gira continuamente: el intervalo entre cada impulso determinará la velocidad a la cual gira. Para programar esto sólo es necesario un par de bucles de demora de longitud regulable, dentro de una estructura repetitiva más grande, para determinar el período de tiempo en el que el motor está encendido o apagado durante cada ciclo.

BBC MICRO

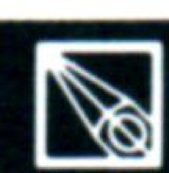
```
10 REM CONTROL DE MOTOR VARIABLE BBC
20 RDD=&FE62:REGDAT=&FE60: VELOCIDAD=30
30 ?RDD=255:REM TODAS SALIDA
40 ?REGDAT=0:REM TODO APAGADO
50 A$=GET$:REM ESPERAR PULSACION DE TECLA
60 REPEAT
70 A$=INKEY$(1)
80 ?REGDAT=0:REM APAGAR
90 FORI=1TO(100-VELOCIDAD):NEXT:REM
   DEMORA1
100 ?REGDAT=1:REM ENCENDER
110 FORI=1TOVELOCIDAD:NEXT:REM DEMORA2
120 IF A$="D" THEN VELOCIDAD=VELOCIDAD-5
130 IF A$="Z" THEN VELOCIDAD=VELOCIDAD+5
140 UNTIL A$="X"
150 ?REGDAT=0:REM APAGAR
```

COMMODORE 64

```
10 REM CONTROL DE MOTOR VARIABLE CBM64
20 RDD=56579:REGDAT=56577:VELOCIDAD=30
30 POKERDD,255:REM TODAS LAS LINEAS SALIDA
40 POKEREGDAT,0:REM APAGAR
50 GETA$:IFA$=""THEN50:REM ESPERAR
   PULSACION TECLA
60 GETA$
70 POKEREGDAT=0:REM APAGAR
80 FORI=1TO(100-VELOCIDAD):NEXT:REM
   DEMORA1
```



Paseo sobre cuatro ruedas
Las dos cajas comparten el bus de datos y la fuente de alimentación eléctrica a través de sus puertas minicon; si cada caja se construye con un conector y un enchufe, entonces se pueden conectar cajas en cadena.
El coche tiene un motor de 12 v de continua, suministrados desde la caja de salida y conmutado mediante el bit 7 de la caja buffer. La polaridad de las salidas se pueden invertir por software, activando el coche hacia atrás y hacia adelante



```
90 POKEREGDAT=1:REM ENCENDER
100 FORI=1TOVELOCIDAD:NEXT:REM DEMORA2
110 IFA$="D"THENVELOCIDAD=VELOCIDAD-5
120 IFA$="Z"THENVELOCIDAD=VELOCIDAD+5
130 IFA$ <> "X"THEN60
140 POKEREGDAT,0:REM APAGAR
```

En este programa se utiliza la variable VELOCIDAD para determinar la longitud de cada bucle de demora. La lógica del bucle es tal que a medida que una demora aumenta, la otra disminuye, y viceversa. DEMORA1 determina el período en el cual el motor está apagado, y DEMORA2 el período en el cual está encendido. Para grandes valores de VELOCIDAD, la primera demora es breve y la segunda es larga, haciendo que el motor gire más rápidamente. Valores más pequeños de VELOCIDAD producirán períodos más largos con el motor apagado durante cada ciclo, haciendo que parezca que gire más lentamente. El efecto de impulso del programa se puede observar en el parpadeo del LED de la línea 1.



Diversión y aprendizaje
Conducir coches de juguete guiados por cable no parece ser una gran compensación para el esfuerzo y el capital invertidos, pero la construcción y programación de incluso estos artefactos tan simples nos ha introducido en la realidad de la construcción electrónica y electromecánica y nos ha mostrado algunos de los problemas a resolver para lograr la interacción entre el software y el mundo real

Ian McKinnell

Respuestas a los ejercicios

1) Suponiendo que los sensores están conectados a las líneas 6 y 7, y que el motor está conectado entre los terminales positivos de las líneas 0 y 1:

```
10 REM BBC VERSION 3.1
20 RDD=&FE62:REGDAT=&FE60
30 ?RDD=63:REM LINEAS 6 & 7 ENTRADA
40 adelante=1:atras=2
50 ?REGDAT=adelante
60 FORI=1TO2000:NEXT:REM DEMORA
70 REPEAT UNTIL(?REGDAT AND192) <> 192
80 ?REGDAT=atras
90 FORI=1TO2000:NEXT:REM DEMORA
100 IF(?REGDAT AND192) <> 192 THEN50 ELSE
    GOTO100

10 REM CBM 64 VERSION 3.1
20 RDD=56579:REGDAT=56577
30 POKERDD,63:REM LINEAS 6&7 ENTRADA
40 AD=1:AT=2
50 POKE REGDAT,AD
60 FORI=1TO1000:NEXT:REM DEMORA
70 IF(PEEK(REGDAT)AND192)=192THEN70
80 POKE REGDAT,AT
90 FORI=1TO1000:NEXT:REM DEMORA
100 IF(PEEK(REGDAT)AND192) <> 192THEN50
110 GOTO100
```

2) Suponiendo que el sensor se conecta a la línea 7 y el motor se conecta entre las líneas 0 y 1:

```
10 REM BBC VERSION 3.2
20 RDD=&FE62:REGDAT=&FE60
30 ?RDD=127:REM LINEA 7 ENTRADA
40 velocidad=30:adelante=1:atras=2
50 direccion=adelante
60 REPEAT
70 ?REGDAT=0:REM APAGADO
80 FOR I=1TO(100-velocidad):NEXT
90 ?REGDAT=direccion
100 FOR I=1TOvelocidad:NEXT
110 UNTIL(?REGDAT AND128)=0:REM INTERRUPTOR
120 FOR I=1TO1000:NEXT:REM DEMORA
130 REM COMPROBAR SI EXCEDIDO
140 IF(?REGDAT AND128)=0THEN?REGDAT=0:END
150 REM ATRAS LENTAMENTE
160 velocidad=2:direccion=atras:GOTO60

10 REM CBM64 VERSION 3.2
20 RDD=56579:REGDAT=56577
30 POKE RDD,127:REM LINEA 7 ENTRADA
40 VEL=30:AD=1: AT=2
50 DR=AD
60 POKE REGDAT,0:REM APAGADO
70 FOR I=1TO(100-VEL):NEXT
80 POKE REGDAT,DR
90 FORI=1TOVEL:NEXT
100 IF(PEEK(REGDAT)AND128) <> 0THEN60
110 FORI=1TO1000:NEXT:REM DEMORA
120 IF(PEEK(REGDAT)AND128)=0THENPOKE
    REGDAT,0:END
130 REM HACIA ATRAS LENTAMENTE
140 VEL=2:DR=AT:GOTO60
```

3) Suponiendo que los sensores de 40 y 70° están conectados a las líneas 6 y 7 respectivamente, y que el calentador está fijado a la línea 0:

```
10 REM BBC VERSION 3.3
20 RDD=&FE62:REGDAT=&FE60
30 ?RDD=63:REM LINEAS 6&7 ENTRADA
40 REPEAT
50 A$=INKEY$(1)
60 ?REGDAT=1:REM ENCENDER CALENTADOR
70 REPEAT
80 UNTIL(?REGDAT AND192)=0:REM 70 GR
90 ?REGDAT=0:REM APAGAR CALENTADOR
100 REPEAT UNTIL(?REGDAT AND192)=192
110 UNTIL A$ <> "":REM PULSACION TECLA

10 REM CBM64 VERSION 3.3
20 RDD=56579:REGDAT=56577
30 POKE RDD,63:REM LINEAS 6&7 ENTRADA
40 GET A$
50 ?REGDAT=1:REM ENCENDER CALENTADOR
60 IF(PEEK(REGDAT)AND192) <> 0THEN60
70 POKE REGDAT,0:REM APAGAR CALENTADOR
80 IF(PEEK(REGDAT)AND192) <> 192THEN80
90 IF A$=""THEN40
```

4) Suponiendo que el primer interruptor está en la línea 6 y el segundo en la línea 7:

```
10 REM BBC VERSION 3.4
20 RDD=&FE62:REGDAT=&FE60:DISTANCIA=1
30 ?RDD=63
40 REPEAT UNTIL(?REGDAT AND64)=0
50 ?REGDAT=1
60 TIEMPO=0:REM INICIALIZAR TEMPORIZADOR
70 REPEAT UNTIL(?REGDAT AND128)=0
80 PRINT"VELOCIDAD="DISTANCIA/(TIEMPO/100)
90 ?REGDAT=0

10 REM CBM64 VERSION 3.3
20 RDD=56579:REGDAT=56577:DS=1
30 POKE RDD,63
40 IF(PEEK(REGDAT)AND64) <> 0THEN40
50 POKE REGDAT,1
60 T=TI:REM INICIALIZAR TEMPORIZADOR
70 IF(PEEK(REGDAT)AND128) <> 0THEN60
80 PRINT"VELOCIDAD="DS/((TI-T)/60)
90 POKE REGDAT,0
```

Formación de caracteres

A través de dos programas le mostramos con qué facilidad se pueden crear caracteres en el Spectrum y el BBC Micro

El enfoque del Spectrum a los caracteres definidos por el usuario es muy práctico: hay 168 bytes en el tope de la memoria dedicados por el sistema operativo a las definiciones de caracteres del usuario (si bien esta memoria se puede utilizar para otros fines). Este espacio permite almacenar 21 definiciones de caracteres, y el sistema operativo le adjunta a cada definición un valor CHR$ en la escala entre 144 y 164. Los caracteres así definidos son considerados por la máquina como los caracteres de la "a" a la "u" en modalidad de gráficos (G). La variable del sistema (dirección 32600) para gráficos definidos por el usuario (UDG) apunta al primer byte de la zona de memoria dedicada, pero no es necesario efectuar sumas complicadas con esta variable para hallar la dirección de comienzo de una definición de carácter. La instrucción LET DIRECCION=USR"A" devuelve la dirección del primer byte de la definición del carácter contenido entre las comillas.

Comparando la longitud y la estructura de los listados para el Commodore y el Spectrum se aprecia la sencillez de los métodos de este último. El programa es una traducción de la versión para el Commodore (p. 1052). Las teclas del cursor normales del Spectrum (SHIFT 5, SHIFT 6, etc.) controlan el cursor de edición, y las teclas 6, 7 y 8 sin SHIFT son de instrucciones (activar una celda, cambiar el carácter editado y colocar un carácter en la ventana de texto). El signo de exclamación es la instrucción de salida.

Spectrum

Una vez definidos los nuevos caracteres, puede guardarse (SAVE) el área UDG con la instrucción:

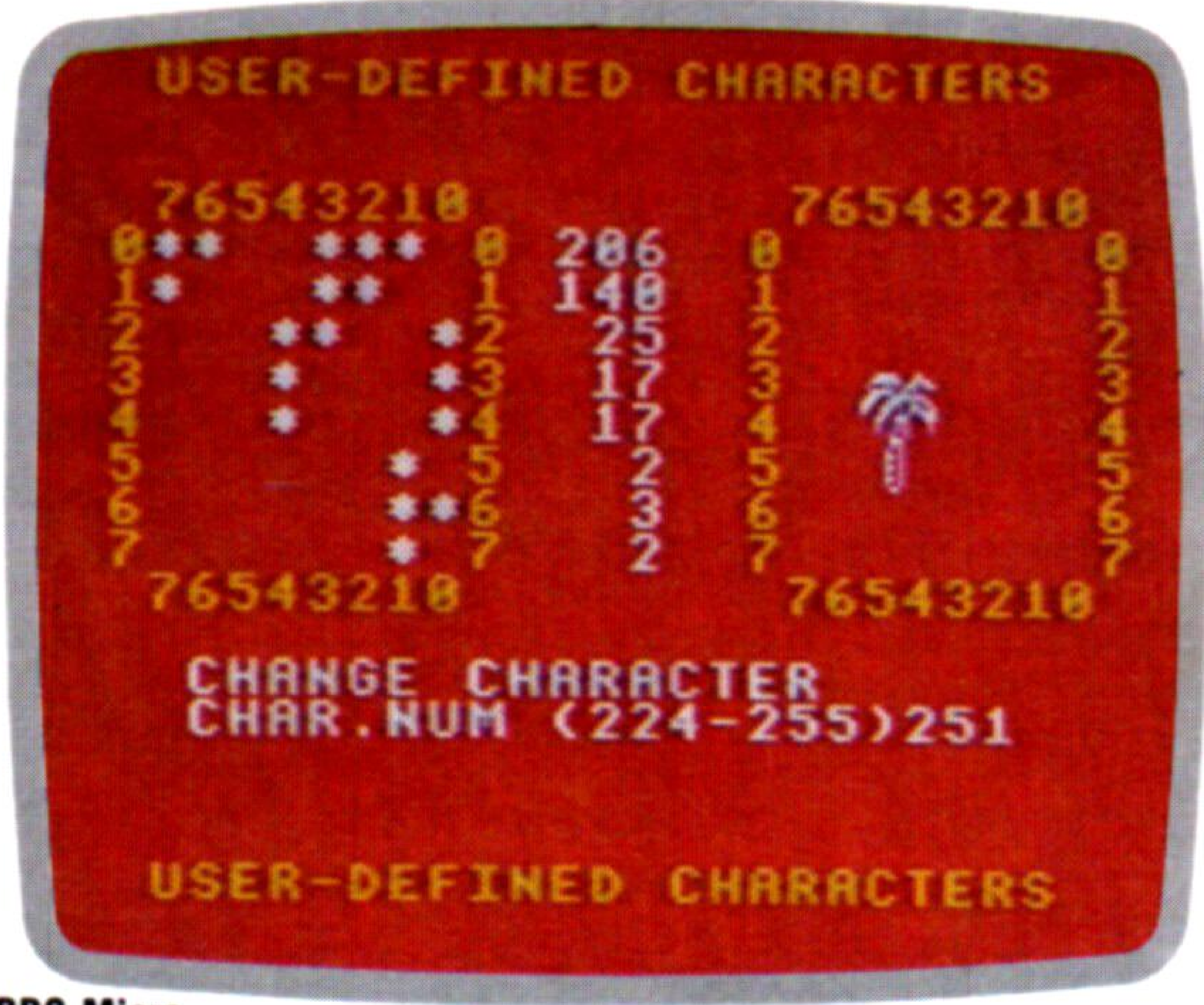
```
SAVE"archudg" CODE(USR"A"),168
```

y volver a cargarla con:

```
LOAD"archudg"CODE
```

El tratamiento del BBC de los gráficos definidos por el usuario es similar, a primera vista, al del Spectrum. Los 256 bytes entre &0C00 y &0CFF están reservados para las definiciones de los caracteres codificados en ASCII del 224 al 255. Si el juego de caracteres se implosiona (véase *Guía para el usuario avanzado del BBC*, p. 136), estas definiciones también se aplican a los códigos ASCII del 128 al 159, del 160 al 191 y del 192 al 223. En el estado explosionado, todo el juego de caracteres imprimibles (desde CHR$(32) hasta CHR$(255)) puede ser redefinido, pero al costo de RAM del usuario. Para la mayoría de los fines, debería ser suficiente un máximo de 32 caracteres especiales.

Las funciones del programa son las mismas que en las versiones para el Commodore y el Spectrum. Las teclas de flecha controlan el cursor, y las teclas de función, de f1 a f3, las instrucciones para activar, redefinir y colocar. Como antes, el signo de exclamación, "!", es el terminador del programa.



BBC Micro

Ian McKinnell

Observe las llamadas al sistema operativo de las líneas 70 a 180: activan y desactivan las funciones COPY de modo que se puedan utilizar en el programa las teclas de flecha; si se sale del programa utilizando un modo diferente a la instrucción "!", debe digitarse *FX4,0 al comienzo de una nueva línea para reactivar las teclas de edición.

El BBC Micro permite emplear VDU23 para definir caracteres, pero en este caso es más simple calcular las direcciones pertinentes y después cogerlas (PEEK) y colocarlas (POKE) individualmente.

Puede salvar (SAVE) sus caracteres especiales así:

```
*SAVE"nombrearchivo"0C00 0CFF
```

y volver a cargarlos con:

```
*LOAD""
```



Spectrum

```
  19 REM ******************************
  20 REM*            spectrum           *
  21 REM *gen.de car.def.usuario*
  22 REM ******************************
 100 GO SUB 1000: REM inicializar
 110 FOR j=0 TO 1 STEP 0
 120 GO SUB 2500: REM entrada
 130 GO SUB 3000: REM validar
 140 GO SUB apuntador:REM obedecer
 150 NEXT j
 200 STOP
 999 REM ******************************
1000 REM *           inicializar        *
1001 REM ******************************
1020 DIM b(8,8): DIM c$(2,2): DIM o(2,7): DIM
     r$(8): DIM d$(1): DIM t(8,8)
1100 LET teclanula=2000: LET movercrsr
     =3500: LET instruccion=3000: LET
     actualizar=6500
1200 REM *********inicial.pantalla *********
1220 LET r0=4: LET c0=3: LET r1=8: LET
     c1=8: LET of=16
1230 LET negro=0: LET azul=1: LET cyan=5:
     LET blanco=7
1240 PAPER cyan: INK negro
1250 LET z$="Caracteres definidos por el
     usuario":PRINT AT 1,4;z$; PRINT AT
     20,4;z$
1260 LET z$="  76543210": PRINT AT
     r0,c0;z$;: PRINT TAB c0+of;z$
1270 PRINT AT r0+c1+1,c0;z$;: PRINT TAB
     c0+of;z$
1280 FOR r=r0+1 TO r0+c1
1290 LET z$=STR$ (r-r0-1): LET
     z$=z$+"          "+z$
1300 PRINT AT r,c0;z$;AT r,c0+of;z$
1310 NEXT r
1420 REM ********* despl. cursor *********
1440 DATA -1,+1,0,0
1445 DATA 0,0,+1,-1
1450 DATA 0,0,+1,-1
1460 FOR k=1 TO 2:FOR l=1 TO 4
1470 READ o(k,1):NEXT l:NEXT k
1600 REM ****** inicial.ventana texto*******
1620 FOR r=1 TO r1
1640 FOR c=1 TO c1
1660 LET t(r,c)=32
1680 NEXT c:NEXT r
1800 LET rpos=1: LET cpos=1:LET cn=144:
     LET e$="A"
1820 GO SUB 6000
1990 RETURN
1999 REM ******************************
2000 REM *    pulsacion tecla invalida    *
2001 REM ******************************
2020 BEEP .2,-3: RETURN
2120 RETURN
2499 REM ******************************
2500 REM *        crsr @,rpos,cpos        *
2501 REM ******************************
2520 LET rp=r0+rpos: LET cp=c0+cpos: LET
     cf=1+b(rpos,cpos): LET d$="*"(cf)
2540 PRINT AT rp,cp; FLASH 1;d$
2600 REM LET t$=INKEY$:IF t$ <> "" THEN
     GO TO 2600
2620 LET t$=INKEY$: IF t$="" THEN GO TO
     2620
2640 PRINT AT rp,cp;FLASH 0;d$
2700 RETURN
2999 REM ******************************
3000 REM *         validar entrada        *
3001 REM ******************************
3020 IF t$="!" THEN LET apuntador=teclanula:
     LET j=2: RETURN
3040 LET tecla=CODE (t$)-7: LET k=tecla-45:
     LET apuntador=teclanula
3060 IF (tecla > =1 AND tecla < =4) THEN
     LET apuntador=movercrsr: RETURN
3080 IF(t$ > "6" AND t$ < ="8") THEN LET
     tecla=VAL (t$)-4: LET apuntador=
     instrucción+tecla*500: RETURN
3100 RETURN
3499 REM ******************************
3500 REM *         mover el cursor        *
3501 REM ******************************
3520 LET ny=rpos+o(2,tecla): LET
     nx=cpos+o(1,tecla)
3540 IF (ny < 1 OR ny > r1) THEN RETURN
3560 IF (nx < 1 OR nx >r1) THEN RETURN
3580 LET rpos=ny: LET cpos=nx
3600 RETURN
3999 REM ******************************
4000 REM *       activar una celda        *
4001 REM ******************************
4020 LET tg=1-b(rpos,cpos): LET
     d$="*"(1+tg)
4040 PRINT AT rpos+r0,cpos+c0;d$
4060 LET b(ros,cpos)=tg
4120 LET pe=PEEK (mpos+rpos)
4140 LET pe=pe+(tg*2-1)*(2^(8-cpos))
4160 POKE (mpos+rpos),pe
4200 PRINT AT r0+rpos,c0+c1+3;PEEK
     (mpos+rpos);" "
4220 NEXT r
4300 RETURN
4499 REM ******************************
4500 REM *     definir nuevo caracter     *
4501 REM ******************************
4520 PRINT AT r0+r1+r3,c0+2;FLASH
     1;"Cambiar caracter"
4530 GO SUB actualizar
4540 PRINT AT r0+r1+4,c0+2; "Pulse tecla
     (a-u)"
4550 FOR z=1 TO 1
4560 LET e$=INKEY$: IF e$ <>"" THEN GO TO
     4560
4570 LET e$=INKEY$: IF e$="" THEN GO TO
     4570
4580 IF (e$ < "a" OR e$ > "u") THEN GO SUB
     teclanucla: LET z=0
4590 NEXT z
4600 LET cn=CODE (e$)+79
4610 IF cn > 164 THEN LET cn=cn-32
4620 PRINT AT r0+r1+3,c0+2;FLASH
     0;"                "
4630 PRINT AT
     r0+r1+4,c0+2;"
4640 GO SUB 6000
4990 RETURN
4999 REM ******************************
5000 REM *       situar un caracter       *
5001 REM ******************************
5020 PRINT AT rpos+r0,cpos+c0+of;CHR$ (cn)
5040 LET t(rpos,cpos)=cn
5490 RETURN
5999 REM ******************************
6000 REM *        visualizar un car.      *
6001 REM ******************************
6020 LET mpos=USR (e$)-1: INK azul
6040 FOR r=1 TO c1: LET pe=PEEK(mpos+r):
     LET p=pe: LET z$=""
6060 FOR c=c1 TO 1 STEP -1: LET n=INT
     (pe/2): LET q=pe-2*n
6080 LET b(r,c)=q: LET pe=n
6100 LET r$(c)=" *" (q+1): NEXT c
6120 PRINT AT r0+r,c0+1;r$;AT
     r0+r,c0+c1+3;p;"  "
6130 NEXT r
6140 RETURN
6499 REM ******************************
6500 REM *    actualizar ventana texto    *
6501 REM ******************************
6520 FOR r=1 TO 8: LET y=r0+r
6540 FOR c=1 TO 8: LET x=c0+c+of
6560 PRINT AT y,x;CHR$ (t(r,c))
6580 NEXT c:NEXT r
6600 RETURN
```

BBC Micro

```
  19 REM  ******************************
  20 REM *          BBC Micro          *
  21 REM  ******************************
  22 REM *   GEN.CAR.DEF.POR USUARIO   *
  50 MODE 1
  60 @ %=3
  70 *FX4,1
 100 PROCinicializar
 110 REPEAT
 120 PROCtomar—instruccion
 130 PROCvalidar—instruccion
 140 PROCobedecer(INSTRUCCION)
 160 UNTIL INSTRUCCION=SALIR
 180 *FX4,0
 200 END
 999 REM  ******************************
1000 DEPROCinicializar
1001 REM  ******************************
1040 DIM BD(8,8),C$(2,2)
1045 DIMOFST(2,7),D$(1),TX(8,8)
1060 D$(0)=" ":D$(1)="*"
1080 CGEN=&0C00-1
1100 SALIR=-1:TECLANULA=0:MVRCRSR=1
1105 TGGLE=2:DFINIR=3:COLOCAR=4
1200 REM  ***** INICIALIZAR PANTALLA ******
1220 R0=4:C0=3:RL=8:CL=8
1225 C9=CL+3:OF=16
1230 NEGRO=0:ROJO=1:AMARILLO=2
1235 BLANCO=3:FONDO=128
1240 COLOUR FONDO+ROJO
1245 COLOUR AMARILLO:CLS
1250 Z$="CARACT. DEF. POR USUARIO"
1255 PRINTTAB(4,1)Z$;TAB(4,20)Z$
1260 Z$="  76543210"
1264 PRINTTAB(C0,R0)Z$
1268 PRINTTAB(C0+OF,R0)Z$
1270 Y=R0+RL+1:X=C0
1274 PRINTTAB(X,Y)Z$
1278 PRINTTAB(X+OF,Y)Z$
1280 FOR R=R0+1 TO R0+RL
1290 Z$=STR$(R-R0-1)
1295 Z$=Z$+"         "+Z$
1300 PRINTTAB(C0,R)Z$
1304 PRINTTAB(C0+OF,R)Z$
1310 NEXT R
1320 COLOUR BLANCO
1420 REM  ******** DESPL. CURSOR *********
1440 DATA -1,+1,0,0
1450 DATA 0,0,+1,-1
1460 FOR K=1 TO 2:FOR L=1 TO 4
1470 READ OFST(K,L):NEXT L,K
1500 REM  ** ESTABLECIMIENTO DE TECLAS ***
1520 *TECLA1"T"
1540 *TECLA2"D"
1560 *TECLA3"P"
1600 REM  ***** INICIAL. VENTANA TEXTO *****
1620 FOR R=1 TO RL
1640 FOR C=1 TO RL
1660 TX(R,C)=32
1680 NEXT C,R
1800 RPOS=1:CPOS=1:CN=224
1805 PROCvisualizar—caracter
1990 ENDPROC
1999 REM  ******************************
2000 DEFPROCteclanula
2001 REM  ******************************
2020 SOUND 1,-15,48,10
2120 ENDPROC
2499 REM  ******************************
2500 DEFPROC tomar—instruccion
2501 REM  ******************************
2520 RP=R0+RPOS:CP=C0+CPOS
2540 PRINTTAB(CP,RP);
2620 T$=GET$
2700 ENDPROC
2999 REM  ******************************
3000 DEFPROC validar—instruccion
3001 REM  ******************************
3020 INSTRUCCION=MVRCRSR
3040 TECLA=ASC(T$)-135
3060 IF TECLA < 1 THEN
     INSTRUCCION=TECLANULA
3065 IF TECLA > 4 THEN
     INSTRUCCION=TECLANULA
3080 IF T$="T" THEN INSTRUCCION=TGGLE
3100 IF T$="D" THEN INSTRUCCION=DFINIR
3120 IF T$="P" THEN INSTRUCCION=COLOCAR
3150 IF T$="!" THEN INSTRUCCION=SALIR
3160 ENDPROC
3299 REM  ******************************
3300 DEFPROCobedecer(instruccion)
3301 REM  ******************************
3320 IF instruccion=SALIR THEN PROCteclanula
3340 IF instruccion=MVRCRSR THEN
     PROCmovercursor(TECLA)
3360 IF instruccion=TGGLE THEN
     PROCactivar—una—celda
3380 IF instruccion=DFINIR THEN
     PROCdefinir—caracter
3400 IF instruccion=COLOCAR THEN
     PROCcolocar—caracter
3420 IF instruccion=TECLANULA THEN
     PROCteclanula
3450 ENDPROC
3499 REM  ******************************
3500 DEFPROCmovercursor(tecla)
3501 REM  ******************************
3520 NY=RPOS+OFST(2,tecla)
3525 NX=CPOS+OFST(1,tecla)
3540 IF NY > =1 AND NY < =RL THEN RPOS=NY
     ELSE PROCteclanula
3560 IF NX > =1 AND NX < =CL THEN CPOS=NX
     ELSE PROCteclanula
3600 ENDPROC
3999 REM  ******************************
4000 DEFPROCactivar—una—celda
4001 REM  ******************************
4020 TG=1-BD(RPOS,CPOS)
4025 Y=RPOS+R0:X=CPOS+C0
4040 PRINTTAB(X,Y)D$(TG)
4060 BD(RPOS,CPOS)=TG
4120 PE=?(MPOS+RPOS)
4140 PE=PE+(TG*2-1)*(2^(8-CPOS))
4160 ?(MPOS+RPOS)=PE
4200 PRINTTAB(C0+C9,R0+RPOS)PE
4220 PROCactualizar—texto
4300 ENDPROC
4499 REM  ******************************
4500 DEFPROCdefinir—caracter
4501 REM  ******************************
4510 REPEAT
4520 PRINTTAB(C0+2,R0+RL+3);
4525 PRINT"                    "
4540 PRINTTAB(C0+2,R0+RL+4);
4545 PRINT"                    "
4550 PRINTTAB(C0+2,R0+RL+4);
4560 INPUT"NUMERO (224-255)" CARNUM
4580 UNTIL CARNUM > 223 AND CARNUM < 256
4600 CN=CARNUM
4620 PROCvisualizar—caracter
4900 ENDPROC
4999 REM  ******************************
5000 DEFPROCcolocar—caracter
5001 REM  ******************************
5015 X=CPOS+C0+OF:Y=RPOS+R0
5020 PRINTTAB(X,Y);CHR$(CN)
5040 TX(CPOS,RPOS)=CN
5400 ENDPROC
5999 REM  ******************************
6000 DEFPROCvisualizar—caracter
6001 REM  ******************************
6020 MPOS=(CN-224)*8+CGEN
6040 FOR R=1 TO CL
6045 PE=?(MPOS+R):P=PE:Z$=""
6060 FOR C=CL TO 1 STEP -1
6065 N=INT(PE/2):Q=PE-2*N
6080 BD(R,C)=Q:PE=N
6100 Z$=D$(Q)+Z$:NEXT C
6120 PRINTTAB(C0+1,R0+R)Z$
6125 PRINTTAB(C0+C9,R0+R)P
6130 NEXT R
6140 ENDPROC
6499 REM  ******************************
6500 DEFPROCactualizar—texto
6501 REM  ******************************
6520 FOR R=1 TO RL:Y=R0+R
6540 FOR C=1 TO RL:X=C0+C+OF
6560 PRINTTAB(X,Y)CHR$(TX(C,R))
6580 NEXT C,R
6900 ENDPROC
```

Rayos eficaces

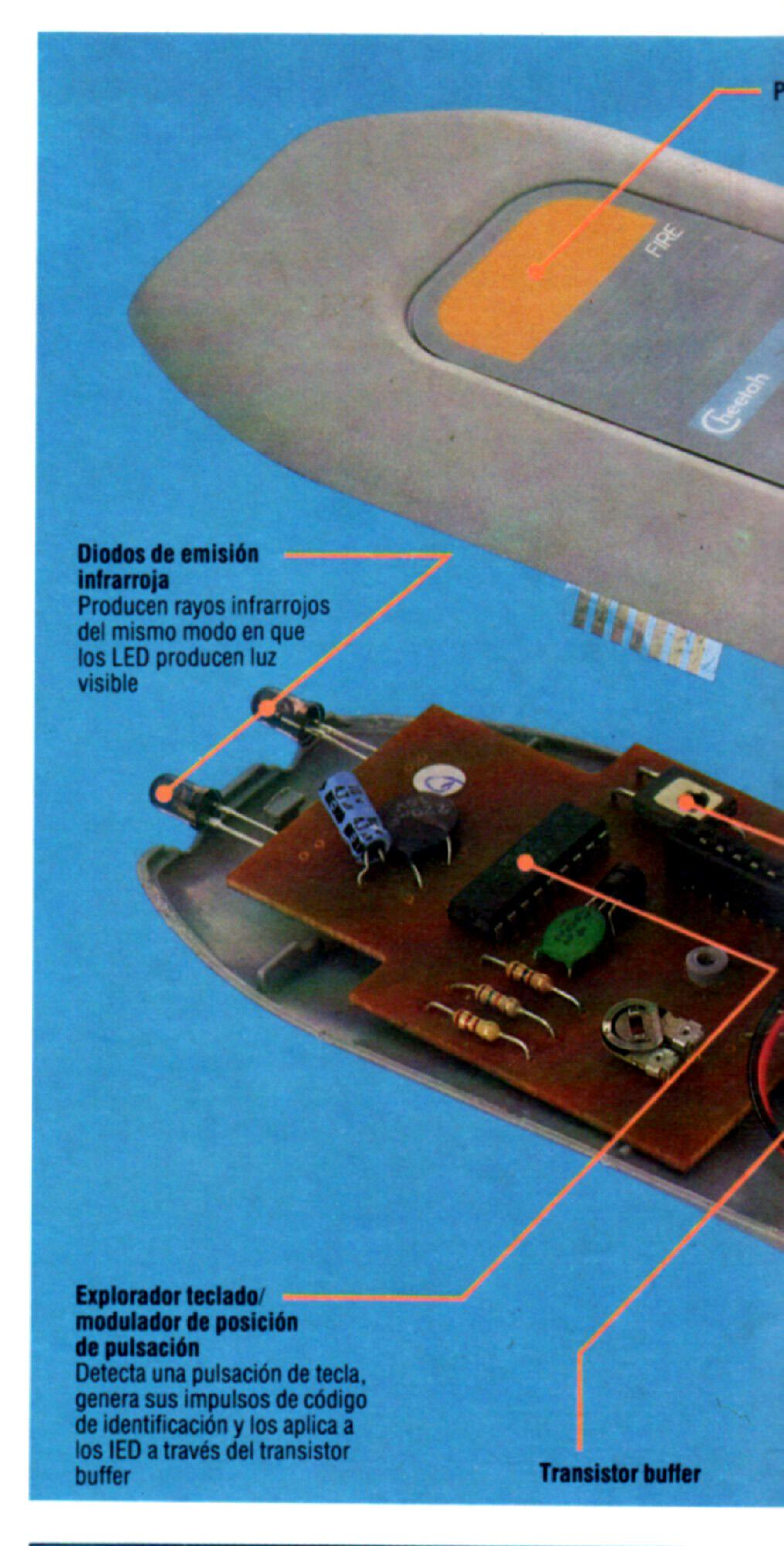
Presentamos la RAT, una palanca a control remoto para el Spectrum que utiliza ondas infrarrojas

Los buenos aficionados a los juegos recreativos tienden a preocuparse mucho por cualquier cosa que incida significativamente en la velocidad y la calidad de su juego, en especial si también ejerce un impacto considerable en el marcador final. Por este motivo, la forma en que se controla un juego es una cuestión de gran importancia. Con los juegos controlados desde el teclado, las principales cuestiones son la selección de teclas y la facilidad con que las mismas se puedan utilizar. Por consiguiente, los escritores de software deben prestar especial atención a esta clase de detalles. Con los controladores externos, como las palancas de mando, el diseño de hardware tiende a constituir el factor esencial.

La flexibilidad de una palanca de mando (su libertad de movimiento, la rapidez con que reacciona al tacto, la velocidad a la cual responde el juego) debería ser su característica de diseño más significativa. Pero con frecuencia el diseñador se encuentra trabado por las limitaciones impuestas por la naturaleza de la palanca: la longitud del cable de conexión, el tamaño, la forma y la posición del controlador, y la posición del o de los pulsadores de disparo. Este último detalle, por ejemplo, suele favorecer a los jugadores diestros. A pesar de que los fabricantes de palancas de mando han intentado desarrollar diseños que obvien estos inconvenientes, ninguno lo ha conseguido con tanto éxito como la palanca a control remoto por rayos infrarrojos de Cheetah Marketing para el Spectrum.

A su palanca Cheetah la ha denominado RAT. Se dice que el nombre corresponde a las siglas de *remote action transmitter* (transmisor de acción remota), pero más bien parece ser un juego de palabras con la palabra "ratón", que se aplica a los controladores de mano derecha utilizados con el Macintosh de Apple y otras máquinas. La RAT (rata) se parece a un arma láser ligeramente más alargada que la que se ve en la serie de televisión *Star trek* (Viaje a las estrellas). Es larga, plana y gris, con un mando circular para control de color azul, el logotipo de Cheetah y un pulsador de disparo de color naranja brillante. Por el frente de la unidad sobresalen dos transmisores infrarrojos. Cuando uno coge la RAT por primera vez y acciona el pulsador de disparo, casi espera ver saltar de ella rayos de llamaradas azules.

El sistema también incluye su propia interface, que se enchufa en el conector marginal de la parte trasera del Spectrum. Esta caja tiene una puerta para ampliación propia para otros accesorios. En el

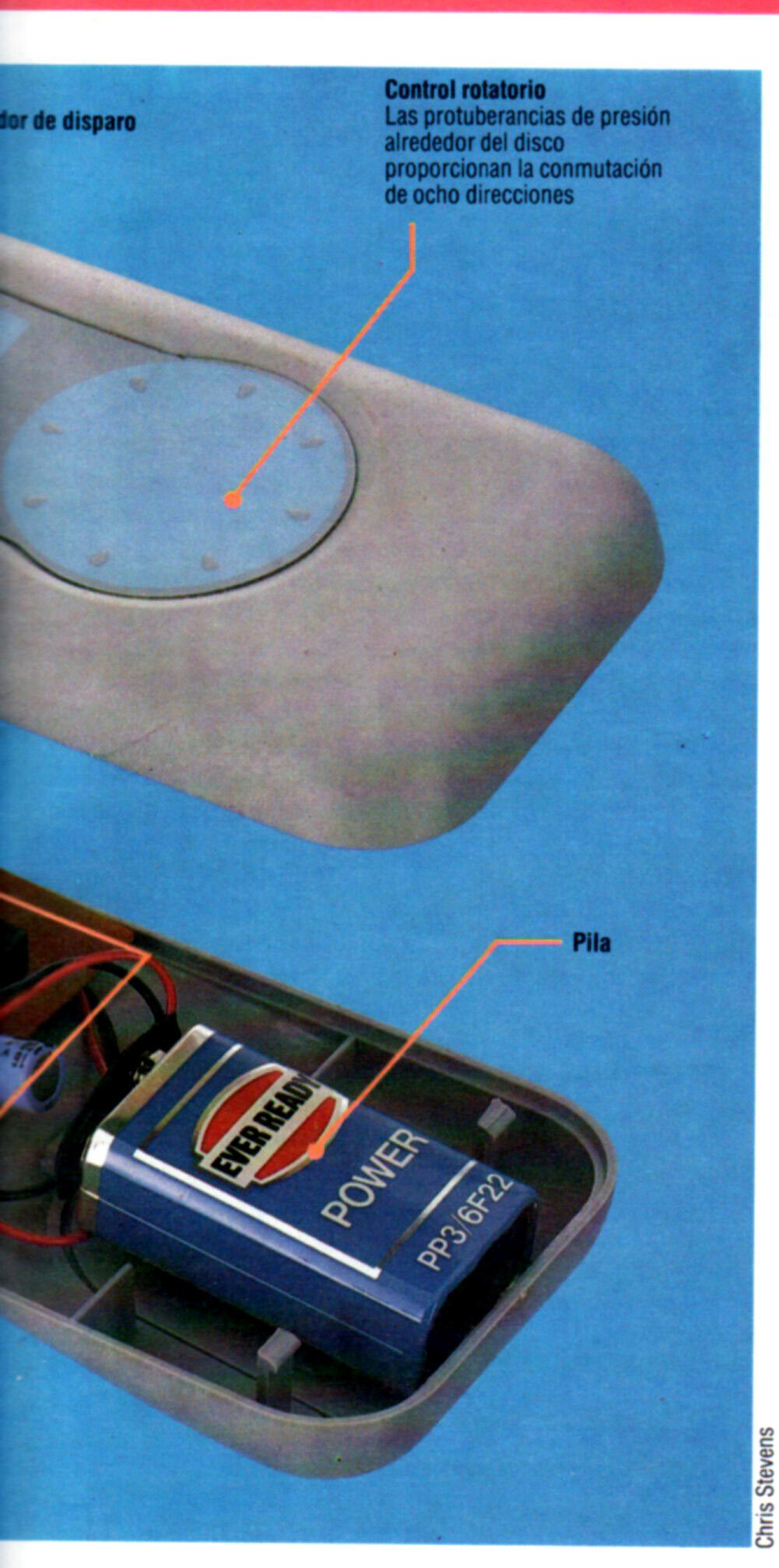




Chris Stevens

Irradiando luz
La radiación infrarroja (que posee mayor longitud de onda que la luz visible pero menor que las ondas de radio) se produce en el transmisor mediante IED (*Infrared Emitting Diodes*: diodos de emisión infrarroja) cuando una corriente eléctrica de un diminuto chip de arseniuro de galio excita las moléculas provocando la liberación de fotones. En el receptor, por el contrario, fluye una corriente eléctrica en el IED cuando la luz infrarroja cae sobre el arseniuro de galio. Por consiguiente, cuando se pulsan los botones de control de la RAT, los dos IED emiten impulsos codificados de infrarrojos en un ancho haz, activando el receptor directamente o después de la reflexión en la habitación





Chris Stevens

Recepción mezclada
El teclado original del IBM PC Junior, tanto tiempo esperado, se distinguió fundamentalmente por la poca calidad de su aspecto e ingeniería; pero fue, asimismo, el primer microordenador con enlace infrarrojo entre teclado y procesador

frente de esta unidad hay un único receptor infrarrojo para comunicación con la RAT.

El paquete viene con una hoja de instrucciones en la que se explica cómo se utiliza la palanca y a qué juegos se puede jugar con ella (todo software que sea compatible con la palanca Kempston). Haciendo un gran alarde de previsión, Cheetah ha incluido rutinas en BASIC y código máquina que permiten incorporar control por RAT en los juegos.

En la hoja de instrucciones se afirma que la RAT se puede utilizar a distancias de hasta 3,5 m apuntando "en la dirección general del ordenador". El movimiento se produce pulsando ligeramente el mando azul de control. En la periferia del mando hay ocho pequeñas protuberancias, y pulsando encima o cerca de ellas se indica la dirección requerida (N, SO, etc., como las direcciones de una brújula). Mientras con una mano se sostiene la RAT y se controla la dirección del movimiento en la pantalla, se puede utilizar la otra para accionar el pulsador de disparo. Debido al diseño de la RAT, el hecho de realizar cada tarea con una u otra mano no supone ninguna diferencia, de modo que la palanca funciona igualmente bien tanto para jugadores diestros como para zurdos. El transmisor requiere una pila PP3, que se coloca en un pequeño espacio en la parte posterior de la unidad directamente debajo del círculo de control azul.

Una vez conectada la caja interface, dotada la unidad transmisora de una pila y cargado en un Spectrum un juego que permita palanca de mando, todo está preparado para ponerse a jugar. Debido a que no hay ninguna señal visible de que el transmisor está funcionando hasta que se ve el movimiento en la pantalla, puede que se sorprenda a sí mismo pegado al ordenador tan cerca como le sucedería con cualquier otra palanca de mando. Existe cierta resistencia a creer que se puede obtener control a una distancia de 3,5 m. Pero cuando uno se da cuenta de que la RAT realmente funciona, entonces desea experimentar para ver desde qué distancia máxima la puede manejar.

De hecho, el transmisor de acción remota funciona sumamente bien a distancias que incluso se acercan a los 4 m. Y no necesariamente hay que tenerlo apuntando hacia la dirección del ordenador. La RAT funciona cuando está orientada directamente hacia el techo, hacia el suelo, de espaldas al usuario o a su lado (si bien le resultará algo difícil ver lo que sucede cuando el transmisor está orientado en ángulos extraños). Evidentemente, la RAT de Cheetah le proporciona al jugador una notable libertad de movimientos. Sin embargo, el mayor inconveniente es que sólo posee ocho posiciones de movimiento: arriba, abajo, derecha, izquierda y puntos intermedios. Sería mucho mejor disponer de un mayor control.

El hecho de que la RAT no posea partes móviles hace que la unidad sea menos susceptible de romperse o desgastarse que las palancas de mando estándares, lo que la hace más durable. De hecho, viene con una garantía de un año. En cuanto a su precio, la RAT Cheetah cuesta apenas un poco más que la mayoría de las otras palancas de mando más la Interface 2. (Por supuesto, si el usuario ya posee esta última, esto no le será de mucho consuelo.) Pero su uso permite muchísima más libertad de movimiento y un control muchísimo mejor que el de la mayoría de las palancas de mando.

PALANCA INFRARROJA CON TRANSMISOR DE ACCIÓN REMOTA CHEETAH

Para: Sinclair ZX Spectrum

Funciona con: Juegos compatibles con la palanca de mando Kempston o la Cheetah

Producida por: Cheetah Marketing, Ltd, 24 Roy St, London EC1, Gran Bretaña

Métodos de clasificación

Esta vez nos referiremos a dos eficaces procedimientos, aplicables tanto a valores numéricos como alfanuméricos

Los siguientes diagramas muestran la representación gráfica de dos métodos de clasificación válidos tanto para valores numéricos como alfanuméricos. Son los métodos de "burbuja", llamados así por el hecho de que, tras cada pasada, al menos el último elemento de la lista queda ordenado.

Ambos ejemplos cuentan con una parte común, que es la correspondiente a la carga de la tabla con un número determinado de elementos. Éstos podrían estar ya definidos en sentencias DATA, o bien ser entrados uno por uno desde teclado. Esta parte común es la mostrada en la figura 1, que puede unirse a cualquiera de las dos versiones del citado método, la simple de la figura 2, o la representada en la figura 3 y que requiere del uso de FOR...NEXT anidados.



Variaciones LOGO

Los procedimientos LOGO son más flexibles si el usuario es capaz de pasar parámetros de entrada a dichos procedimientos

En LOGO, una palabra (aquí vamos a tomar como ejemplo TAMAÑO) se puede utilizar de tres formas diferentes. Para distinguir entre las mismas, el LOGO emplea tres notaciones distintas: TAMAÑO, :TAMAÑO (que se lee "puntos tamaño"), y "TAMAÑO ("comillas tamaño"). Como ya hemos visto, si el LOGO se encuentra con la palabra TAMAÑO, con ningún signo de puntuación que la preceda, la toma como el nombre de un procedimiento y llevará a cabo la secuencia de instrucciones especificada en la definición TAMAÑO. :TAMAÑO se utiliza para indicar el *valor* retenido en el nombre de la variable; si el LOGO se encuentra con :TAMAÑO recuperará el valor asociado al nombre. "TAMAÑO se emplea para aludir a nombres de variables y procedimientos, pero indica que nos estamos refiriendo al nombre en sí mismo y no a cualquier valor asociado al mismo. Por consiguiente, "TAMAÑO se puede utilizar para hacer referencia a una variable, mientras que :TAMAÑO hace referencia al valor que se le haya asignado a esa variable. (Observe que se ponen comillas antes de la palabra pero no después de la misma.)

El MIT LOGO no es totalmente coherente en cuanto a la utilización de esta notación. Después de las instrucciones EDIT, ERASE y PO se debe escribir el nombre de un procedimiento sin comillas. Por lo tanto, la sintaxis correcta es EDIT CUADRADO, aun cuando CUADRADO no es una llamada al procedimiento CUADRADO, sino sólo el nombre del mismo, y debería, lógicamente, ir precedido por comillas. El LCSI LOGO es más coherente y requiere el empleo de comillas con estas instrucciones.

Para utilizar cualquiera de los procedimientos que hemos definido hasta ahora simplemente digitamos el nombre del procedimiento, de la misma manera en que utilizaríamos instrucciones del LOGO tales como DRAW o HIDETURTLE. Sin embargo, otras instrucciones (FORWARD, p. ej.) necesitan información extra antes de que puedan utilizarse. La palabra FORWARD sola no tiene ningún significado: se le debe asignar un valor para que el LOGO pueda ejecutar la instrucción. Si incluimos nombres de variables, podemos entrar cualquier valor requerido y, por consiguiente, variar el efecto obtenido cuando llamemos al procedimiento.

Tomemos el procedimiento que definimos en un capítulo anterior para dibujar un cuadrado:

```
TO CUADRADO
  REPEAT 4 [FD 50 RT 90]
END
```

Tal como está, este procedimiento dibujará un cuadrado con lados de 50 unidades de longitud. No obstante, sería muchísimo más útil que el cuadrado se pudiera dibujar en cualquier tamaño escogido; para hacer esto, debemos entrar el valor deseado. Para cambiar CUADRADO de modo que acepte una entrada, utilice el editor para reemplazar el valor fijo de 50 por la variable "LADO y agregue :LADO a la línea del título. Ahora nuestro procedimiento tendrá el siguiente aspecto:

```
TO CUADRADO :LADO
  REPEAT 4 [FD :LADO RT 90]
END
```

Ahora, cuando se llame el procedimiento, será necesario darle un valor a la variable "LADO. Pruebe con CUADRADO 40, CUADRADO 10, etc., para ver cómo varía el tamaño del cuadrado.

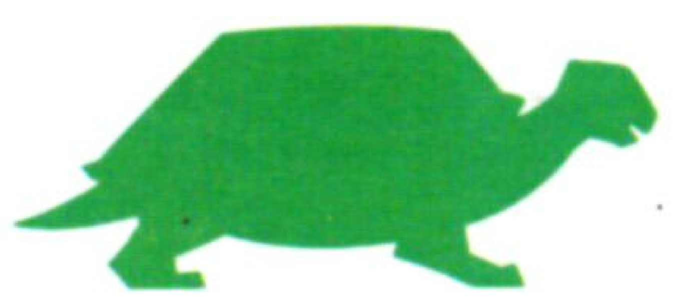
Veamos qué es lo que sucede cuando se digita CUADRADO 30. El LOGO primero busca la información de CUADRADO. La línea del título dice que se requiere una entrada y que la misma se va a llamar "LADO. El valor de la línea de entrada (en este caso, 30) se le asigna a la variable "LADO y entonces se obedecen las instrucciones de la definición del procedimiento. La mejor forma de visualizar esto consiste en imaginar que cada nombre de variable alude a una caja que contiene un valor. Cuando el LOGO llega a la línea FORWARD :LADO va a la caja etiquetada "LADO, recupera el valor que allí encuentre y lo utiliza como la entrada para FORWARD. La caja etiqueta "LADO sólo se emplea en el procedimiento que alude a ella. Si existe otro procedimiento que utilice "LADO como el nombre para una entrada, hará uso de una caja diferente. Por consiguiente, se dice que LADO es una variable *local*.

También podemos utilizar entradas con subprocedimientos. El procedimiento CASA que ofrecemos aquí es nuestra solución al problema que planteamos en el capítulo anterior. Está escrito siguiendo las pautas detalladas anteriormente de manera que se puedan entrar valores diferentes:

```
TO CASA :GRANDE
   CUADRADO :GRANDE
   FD :GRANDE RT 30
   TRI :GRANDE
   LT 30 BK :GRANDE
END
TO CUADRADO :TAMAÑO
   REPEAT 4 [FD :TAMAÑO RT 90]
END
TO TRI :LADO
   REPEAT 3 [FD :LADO RT 120]
END
```

Aquí hemos utilizado tres nombres distintos de variable ("GRANDE, "TAMAÑO y "LADO). Podríamos haberles asignado el mismo nombre, dado que las variables son locales de los procedimientos en que se utilizan, pero ello habría dado lugar a confusión.

Para ver cómo funcionan estos procedimientos, veamos lo que sucede si digitamos CASA 30. El LOGO lee la línea de entrada y le asigna el valor 30 a



la variable "GRANDE de CASA. La primera línea de CASA, por consiguiente, ahora es equivalente a CUADRADO 30. A la variable "TAMAÑO de CUADRADO se le asigna, a su vez, el valor 30. Ahora se ejecuta CUADRADO, con FD :TAMAÑO convirtiéndose en FD 30. Un procedimiento similar se sigue luego cuando se llama a TRI.

Ahora intente adaptar los procedimientos para dibujar el tablero de 5×5 de modo que TABLERO tome como entrada el tamaño del cuadrado.

He aquí un procedimiento que dibuja polígonos, con el número de lados dado como entrada:

```
TO POLI :LADOS
  REPEAT :LADOS [FD 50 RT 360 / :LADOS]
END
```

Utilizando este procedimiento con una entrada, POLI 3 dibujará un triángulo, POLI 4 un cuadrado, y así sucesivamente. Sin embargo, en todos los polígonos que dibuje este procedimiento los lados tendrán 50 unidades de longitud. Un procedimiento más general que dibujara polígonos de cualquier tamaño requeriría dos entradas: una para el número de lados y otra para la longitud requerida de los lados. Para hacer esto, todo cuanto se necesita es adaptar el procedimiento POLI para reemplazar 50 por un nombre de variable y agregarle ese nombre a la línea del título:

```
TO POLI :LADOS :TAMAÑO
  REPEAT :LADOS [FD :TAMAÑO RT 360/:LADOS]
END
```

Ahora POLI 5 30 dibujará un pentágono con lados de 30 unidades de largo. Quizá le interesara adaptar su nueva versión del procedimiento para dibujar un tablero de modo que dibuje un tablero de cualquier número de cuadrados (no sólo de cinco por cinco). Ahora deberá haber dos entradas: la cantidad de cuadrados en cada dirección y su tamaño.

Hasta ahora hemos considerado variables que son locales de los procedimientos que las utilizan. Pero se pueden definir variables que estén disponibles para aplicarlas con todos los procedimientos. Se dice que las mismas son variables *globales* y son útiles para comunicar información entre distintos procedimientos. No obstante, su empleo dificulta la tarea de depuración y, por lo tanto, se las debe utilizar con cuidado.

Para asignarles valores a las variables globales se utiliza la instrucción o comando MAKE. MAKE "LADO 3 le asigna 3 al valor de la variable "LADO. MAKE "LADO :LADO+1 incrementa el valor de "LADO en 1. El significado exacto de la notación en este segundo ejemplo es: hallar el valor de la variable "LADO, sumarle uno, luego asignar el resultado a la variable llamada "LADO. En cada caso, MAKE exige dos entradas: el nombre de la variable y el valor a asignarle a ésta.

Para resumir las características de programación que hemos cubierto en este capítulo del curso de LOGO, hemos diseñado algunos procedimientos para dibujar espirales. El procedimiento principal se llama EQESPI. Éste requiere tres entradas: la longitud inicial de la línea a dibujar, el ángulo que se debe describir en cada esquina de la espiral, y un factor de escala en función del cual se debe multiplicar la longitud inicial para producir el efecto en espiral. Se pueden utilizar distintos juegos de entradas para obtener efectos diferentes: nosotros probamos con 70 283 0.95, 70 143 0.95 y 20 243 1.05. Pruebe con otros juegos de números.

NOWRAP es una nueva instrucción o comando. La misma hace que la tortuga no pueda salir de los márgenes de la pantalla: cuando la tortuga llega al límite de la pantalla el procedimiento se detiene con un mensaje de error "fuera de límites". En muchos casos, el efecto de salir por el margen derecho entrando por el izquierdo inmediatamente puede dar resultados interesantes. En este procedimiento frustraría el efecto de espiral, de modo que se utiliza NOWRAP para desactivarlo.

El procedimiento principal EQESPI dibuja repetidamente una línea (cuya longitud está determinada por el factor de escala), luego gira describiendo un ángulo fijo y por último altera el factor de escala. La longitud de la línea dibujada se incrementa o bien disminuye según el factor de escala sea mayor o menor que 1. El número grande después de REPEAT es simplemente para mantener el procedimiento trabajando durante mucho tiempo. Cuando ya haya visto lo suficiente, pulse Control-G (o Break) para detener la ejecución del procedimiento. La mayoría de las variables son locales, con la excepción de "ESCALA. Ésta es global porque "CRECER cambia su valor y este nuevo valor se debe poner a disposición de S.FORWARD. Por consiguiente, se utiliza "ESCALA para la comunicación entre los dos procedimientos.

Procedimiento espiral

```
TO EQESPI :LADO :ANGULO :FACTOR
  PREPARACION
  REPEAT 1000 [S.FORWARD :LADO DERECHO
      :ANGULO CRECER :FACTOR]
END
TO PREPARACION
  DRAW NOWRAP MAKE "ESCALA1
END
TO CRECER :NUM
  MAKE "ESCALA :ESCALA1 :NUM
END
TO S.FORWARD :DIST
  FORWARD :ESCALA1:DIST
END
```

Complementos al LOGO

Las versiones LCSI utilizan la instrucción FENCE en vez de NOWRAP para impedir el movimiento de la tortuga fuera de los límites de la pantalla.
La versión Atari no posee FENCE, de modo que utilice WINDOW. Ésta evita que la tortuga salga de la pantalla, pero no interrumpe el procedimiento cuando llega al límite de la misma.
En el Spectrum, en Espiral sustituya DRAW por CS en el subprocedimiento PREPARACION.

Procedimientos (2)

1) Escriba un procedimiento para dibujar un círculo de radio 50. Modifique el procedimiento de modo que el radio se dé como entrada al procedimiento.
2) Escriba un procedimiento que dibuje una "diana" con cinco círculos concéntricos.



Respuestas a los ejercicios

1) Puzzles tangram

Las formas del hombre sentado, el hombre haciendo reverencia y el gato utilizan el otro lado de la pieza paralelogramo que empleaba el perro del ejemplo de la semana pasada. Para dar vuelta una pieza en LOGO, simplemente cambie todos los giros DERECHA por IZQUIERDA. De modo que en vez de:

```
TO PAR
   REPEAT 2 [FD 25 RT 45 FD 35 RT 135]
END
```

a veces necesitaremos el "otro lado", dado por:

```
TO PAR1
   REPEAT 2 [FD 25 LT 45 FD 35 LT 135]
END
```

Todos los otros procedimientos de piezas son iguales a los expuestos en el capítulo anterior

Hombre corriendo

```
TO CORRIENDO
   MOVER1 TRI1 MOVER2 PAR MOVER3 TRI3 MOVER4 TRI2
   MOVER5 CUADRADO MOVER6 TRI1 MOVER7 TRI2 MOVER8
END

TO MOVER1
   LT 45
END

TO MOVER2
   PU FD 25 RT 135 FD 17.5 LT 45 PD
END

TO MOVER3
   PU FD 75 RT 90 PD
END

TO MOVER4
   PU RT 90 FD 25 RT 90 FD
END

TO MOVER5
   PU FD 50 RT 135 FD 50 LT 135 PD
END

TO MOVER6
   PU RT 135 FD 21 RT 135 FD 25 LT
   90 FD 50 LT 90 FD 25 RT 90 PD
END

TO MOVER7
   PU FD 25 RT 135 FD 71 RT 15 BK 35 PD
END

TO MOVER8
   PU FD 35 LT 90 FD 25 RT 45 FD
   17.5 LT 45 FD 25 RT 135 FD
END
```

Hombre sentado

```
TO SENTADO
   MOVER1 TRI1 MOVER2 TRI2 MOVER3 TRI3 MOVER4 TRI1
   PAR1 MOVER5 CUADRADO MOVER6 TRI3 MOVER7
END

TO MOVER1
   LT45
END

TO MOVER2
   PU FD 25 LT 45 FD 17.5 RT 90 PD
END

TO MOVER3
   PU BK 15 LT 90 PD
END

TO MOVER4
   PU FD 50 RT 45 FD 25 RT 90 PD
END

TO MOVER5
   PU FD 25 LT 45 FD 35 LT 45 PD
END

TO MOVER6
   PU BK 50 LT 90 PD
END

TO MOVER7
   PU BK 21 RT 135 BK 50 RT 90 FD 35 LT 90 PD
END
```

Hombre haciendo reverencia

```
TO REVERENCIA
   MOVER1 TRI1 MOVER2 PAR1 MOVER3 TRI3 MOVER4 TRI2
   MOVER5 TRI1 MOVER6 TRI2 MOVER7 CUADRADO MOVER8
END

TO MOVER1
   LT 90
END

TO MOVER2
   PU FD 25 RT 135 FD 30 PD
END

TO MOVER3
   PU LT 45 FD 35 LT 135 BK 50 PD
END

TO MOVER4
   PU RT 90 FD 50 LT 135 PD
END

TO MOVER5
   PU RT 90 FD 50 LT 135 FD 5 RT 90 PD
END

TO MOVER6
   PU RT 90 FD 25 RT 45 FD 7.5 RT 45 BK 35 PD
END

TO MOVER7
   PU FD 35 RT 135 FD 7.5 LT 45 FD 55 RT
   45 PD
END

TO MOVER8
   PU LT 45 FD 36 RT 45 FD 56 LT 135 FD 5 LT
   15 BK 25 PD
END
```

Gato

```
TO GATO
   MOVER1 TRI3 MOVER2 CUADRADO MOVER3 TRI1 MOVER4
   TRI1 MOVER5 TRI3 MOVER6 PAR1 MOVER7 TRI2 MOVER8
END

TO MOVER1
   PU FD 50 RT 90 PD
END

TO MOVER2
   RT 170
END

TO MOVER3
   PU RT 90 FD 25 LT 90 PD
END

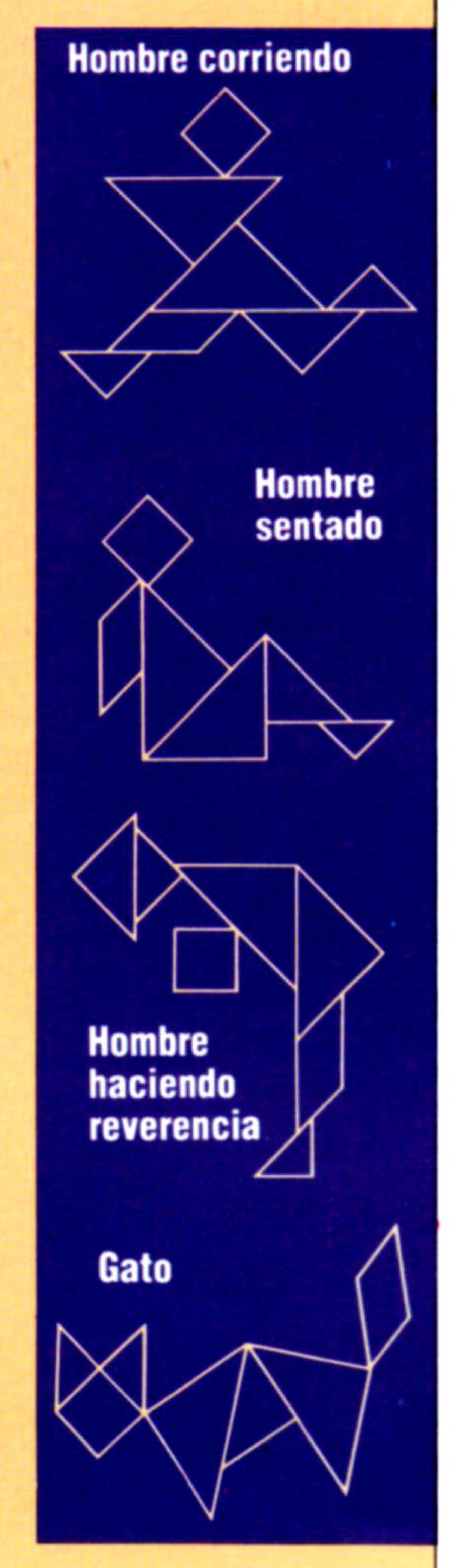
TO MOVER4
   RT 180
END

TO MOVER5
   PU RT 90 FD 25 LT 80 FD 50 RT 45 FD 50 RT 90 PD
END

TO MOVER6
   LT 155
END

TO MOVER7
   PU LT 160 FD 35 PD
END

TO MOVER8
   PU FD 35 LT 45 FD 21 RT 135 PD
END
```



Éstas son nuestras sugerencias para las respuestas a los problemas con procedimientos de la p. 1045:

2)

```
TO CASA
   CUADRADO FD 50
   RT 30 TRI LT 30 BK 50
END

TO CUADRADO
   REPEAT 4 FD 50
   RT 90
END

TO TRI
   REPEAT 3 FD 50
   RT 120
END
```

3)

```
TO TABLERO
   REPEAT 5 LINEA
   FD 10 BK 50
END

TO LINEA
   REPEAT 5 CUADRADO
   RT 90 FD 10 LT 90
   LT 90 FD 50 RT 90
END

TO CUADRADO
   REPEAT 4 FD 10
   RT 90
END
```

4)

```
TO ESTRELLA.SEIS
   TRI MOVER TRI MOVER.
   ATRAS
END

TO TRI
   RT 30 REPEAT 3
   FD 50 RT 120
END

TO MOVER
   PU FD 29 RT 60 PD
END

TO MOVER.ATRAS
   PU LT 60 BK 29 PD
END
```

Rapidez y eficacia

En el presente capítulo consideraremos de qué manera se puede aumentar la velocidad de ejecución de los programas

La programación estructurada y un buen diseño de los programas son técnicas que hacen que éstos sean más fáciles de utilizar, pero no mejoran su eficiencia. Para hacer que los programas funcionen más rápidamente y utilicen menos espacio de memoria, a menudo es necesario sacrificar claridad en el diseño de un programa. De modo que debemos tener presente, al "afinar" un trozo de código, que casi todo lo que se haga por hacerlo más veloz invariablemente hará que también resulte más difícil de leer, de entender y de depurar.

La lentitud inherente a los lenguajes interpretados como el BASIC significa que habrá ocasiones en las que los programas se ejecutarán a una velocidad inaceptable y que se los debe acelerar. La forma más eficaz de acelerar un programa en BASIC es compilarlo. Sin embargo, son muy pocos los micros que soportan un auténtico compilador de BASIC. Existen en el mercado compiladores basados en disco y en cassette, pero la mayoría de ellos sólo soportan BASIC con números enteros y, antes de la compilación, puede que requieran un formateado especial de su programa. La compilación es un proceso lento, especialmente durante el desarrollo del programa y sobre todo cuando el sistema está basado en cassette. El compilador ocupará memoria para el usuario y, cuanto más amplias sean sus facilidades, más RAM requerirá del área para programas del usuario. En general, en los micros personales, la compilación sólo es aconsejable para programas totalmente comprobados y depurados.

Los accesos a archivos son, entre todas las causas, las que más retardan los programas. En un programa que lea y escriba frecuentemente en disco o en cinta (un programa de base de datos, p. ej.), las demoras son inevitables. Acceder a un registro en un archivo de acceso directo o al azar contenido en un disco flexible lleva un promedio de un cuarto de segundo, más o menos. El acceso a datos en archivos secuenciales ocupa más tiempo (y varía según la longitud del archivo) y los accesos a cinta son considerablemente más largos. Si estas demoras están causando problemas, tal vez sea posible reducir la cantidad de accesos leyendo más datos por vez y almacenándolos en RAM, y "economizando" actualizaciones de archivos hasta el final de la sesión. A menudo los programas interactivos originan problemas porque el usuario ha de quedarse contemplando la pantalla durante varios segundos. En este caso, una solución parcial consiste en reorganizar el programa de modo que los archivos se lean y se escriban mientras el usuario está ocupado haciendo alguna otra cosa (leyendo una pantalla llena de instrucciones, p. ej.).

Otra causa de lentitud es la aritmética real. Los números reales son los que tienen posiciones decimales (los enteros son números redondos). Debido a la parte decimal, traer un número real de la memoria y efectuar con él una operación aritmética requiere más ciclos de máquina que hacer lo mismo con un entero. En programas con mucha aritmética, siempre que sea posible conviene reemplazar todas las variables reales involucradas por variables enteras (p. ej., SUM se debería reemplazar por SUM%). Se pueden conseguir economías de alrededor del 20 % incluso para programas moderadamente numéricos, y las aplicaciones de "proceso de números" pueden ganar hasta en un 50 %.

Diseñar un algoritmo más rápido es una de las mejores maneras de acelerar un programa. En este curso ya hemos recomendado algunas fuentes de algoritmos. Pruebe con ellas, y esté atento a las que se publican en las revistas de informática. Por otra parte, diseñar algoritmos es una cuestión de creatividad y perspicacia. Los BASIC por lo general poseen una gran riqueza de funciones incorporadas (como INSTR, SGN, LOG, etc.) que son muy rápidas. Esta velocidad es consecuencia de estar escritas en lenguaje máquina y de utilizar los mejores algoritmos disponibles. Con frecuencia vale la pena volver a examinar el manual para ver qué funciones incorporadas se ofrecen antes de codificar una versión propia. Las funciones definidas por el usuario, implementadas mediante la instrucción DEF FN, también operan velozmente. Esta instrucción es de gran utilidad en programas con cálculos repetidos o una secuencia repetida de manipulaciones de series, donde puede sustituir a una llamada a subrutina, que es mucho más lenta.

Escribir subrutinas en lenguaje máquina generalmente hace que funcionen más rápidamente. Ello se debe a que en los programas interpretados deben traducirse sus instrucciones fuente a código máquina a medida que van siendo encontradas durante la ejecución del programa (tampoco esto se

Compiladores para micros

BBC Micro Turbo Compiler Salamander, 17 Norfolk Road, Brighton BN1 3 AA. 0273 771942	Cassette
Commodore 64 DTL-BASIC Compiler Dataview Wordcraft Ltd., Radix House, East Street, Colchester CO1 2XB. 0206 86914	Cassette/disco
Spectrum Softek FP Softek IS, Combined FP and IS, Softek International, 12/13 Henrietta St, London WC2. 01-240 1422	Cassette Cassette Cassette



hace con particular eficacia). Escribir en lenguaje máquina evita este proceso de traducción. Lamentablemente, escribir programas en lenguaje assembly es mucho más difícil que escribir en BASIC, y puede que el costo en tiempo y esfuerzo no compense la eventual economía. No obstante, algunos programas (los que empleen gráficos animados, p. ej.) no funcionarían como se pretende si estuvieran escritos sólo en BASIC.

Existen muchas otras formas de hacer pequeñas economías en velocidad de proceso. Utilice una variable en vez de un número verdadero (p. ej., MAX en vez de 267,5) para un acceso más rápido a los valores, especialmente en bucles. Utilice letras diferentes para comenzar nombres de variables, y seleccione estas letras iniciales de forma uniforme a través del alfabeto. Emplee líneas de sentencias múltiples (si es posible) y cree un intervalo considerable entre los números de línea (como 10). En los bucles FOR...NEXT, si el intérprete lo permite, deje fuera la variable del contador del bucle (p. ej., utilice NEXT en vez de NEXT LOOP). Dentro de un bucle, intente evitar el tener que calcular el mismo valor una y otra vez. En lugar de eso, calcúlelo fuera del bucle e incorpórelo como una variable.

Ahorrando espacio

La aritmética de enteros no sólo ahorra tiempo sino que también ahorra espacio. Cuando almacenar un número real puede gastar cuatro o cinco bytes, sólo se necesitan dos para almacenar un entero. Esto representa un importante ahorro, especialmente cuando están implicadas grandes matrices. Otras mejoras a la velocidad de un programa contribuirán asimismo a ahorrar espacio: utilizar funciones incorporadas o definidas por el usuario economiza código, al igual que escribir en lenguaje assembly y utilizar líneas de sentencias múltiples. La compilación tiende a aumentar el tamaño de los programas más pequeños y sólo ahorra espacio en los grandes.

La eliminación de sentencias REM representa una evidente economía de espacio, y la utilización de textos cortos para los avisos o mensajes también ayuda. La colocación de grandes bloques de texto en archivos almacenados fuera del programa los quita de en medio cuando los mismos no son necesarios (las mayores cargas son los archivos de instrucciones y de "ayuda"). Elimine de cada línea tantos espacios como sea legal, y emplee números de línea y nombres de variables más cortos. Si se necesita dimensionar una matriz pero no se conoce su tamaño exacto, no se limite a aventurar un número redondeado más o menos conveniente; déjela hasta que tenga la información necesaria y después dimensiónela con una variable, como ésta:

```
10 INPUT "Cuantos casos de esta categoria
   hay?",CASOS %
20 DIM MATRIZ %(CASOS %)
```

Esto se conoce como *dimensionamiento dinámico* y es algo que posee el BASIC y que no ofrecen la mayoría de los otros lenguajes; ¡aprovéchelo!

Otra técnica implica aumentar la memoria destinada al BASIC en RAM. Esto se puede hacer mediante la utilización de instrucciones como HIMEM. Lo que suelen hacer estas instrucciones es cambiar el área de RAM disponible para programas en BASIC y variables. La utilización normal de ésta es para almacenar programas en código máquina en un lugar seguro donde no se los pueda machacar por descuido, pero la misma instrucción se puede emplear para acceder a espacio extra del normalmente reservado para la memoria de pantalla. Si lo que está apareciendo en pantalla no tiene importancia, éste es un buen modo de obtener un kilobyte extra de RAM. Si no es posible modificar HIMEM, la memoria de pantalla todavía se puede utilizar sacando y colocando (PEEK y POKE) directamente en las posiciones de memoria reservadas.

Si todo fracasa y el programa simplemente no cabe en el espacio disponible, muchas versiones de BASIC poseen una instrucción CHAIN que permite que un programa le pase el control a otro. Algunos BASIC permiten la utilización de la instrucción COMMON; ésta le pasa al programa siguiente una serie de variables comunes a ambos y sus valores en curso. En algunos micros, CHAIN (en caso de que exista) suele ser una instrucción muy simple que permite pasar al segundo programa todas o ninguna de las variables del primero.

Si los programas están escritos de forma estructurada, las subrutinas individuales deben aceptar ser escritas y verificadas individualmente. Su ejecución también se puede temporizar de manera individual. Escriba un temporizador sencillo como éste:

```
100 REM Usar esta primera seccion para dar valor a
    las variables
105 REM que necesitara la rutina (no olvidar
110 REM dimensionar matrices y llenarlas con
115 REM datos reales si la rutina utiliza alguno).
120 REM Este programa esta en BASIC BBC y TIME
125 REM es una seudovariable que retiene un valor en
130 REM centesimas de segundo, generado por
135 REM el reloj del sistema
200 COMIENZO=TIME
210 GOSUB 2000:REM Aqui se llama a la rutina que
    se va a temporizar
220 FIN=TIME
230 PRINT "La ejecucion
    llevo";(FIN-COMIENZO)/100;"segundos."
240 END
```

Con esta rutina se pueden experimentar diferentes algoritmos y formas de aumentar la velocidad.

Cómo ser rápido

- Sopese concienzudamente los requerimientos de buen estilo con la necesidad de un código veloz.
- Compile cuando pueda hacerlo; si no le es posible, defina funciones y procedimientos.
- Evite los accesos a archivos.
- Evite los números reales absolutos. Inicialice variables y, si su micro lo permite, emplee aritmética de enteros.
- Diseñe sus algoritmos cuidadosamente y aprenda del ejemplo de otros.
- Considere las ventajas y las desventajas del lenguaje máquina. Si bien puede ser más rápido, consume más tiempo en escritura y depuración.
- Condense su código y elimine sus comentarios REM una vez que tenga una versión operativa.

Cambios de domicilio

En este capítulo vamos a examinar con detalle la utilización de los dos registros índice X e Y para el direccionamiento indexado

La intención primitiva del ordenador con programa almacenado fue la de guardar el programa en el mismo lugar (y de la misma forma) que los datos sobre los que iba a operar para que el programa lograra modificarse a sí mismo según se iba ejecutando. El uso más importante no consiste en modificar las instrucciones sino las direcciones donde están los datos sobre los que actúan esas instrucciones. Imagínese el problema de tener que acceder a varios miles de números teniendo que dar instrucciones individuales para cada uno de ellos si cada instrucción sólo pudiera referirse a una dirección imposible de modificar.

Este problema se pudo resolver en parte gracias al concepto de *modificación de dirección*. Por él una misma instrucción puede repetirse tantas veces como se desee y hacer que se refiera a direcciones diferentes de los datos almacenados, mediante el empleo de un valor cambiante en un registro que altere dicha dirección. Este concepto se usa muy a menudo en BASIC. Por ejemplo:

```
FOR I=1 TO N
PRINT TABLA(I)
NEXT I
```

En este caso una misma instrucción, la de PRINT, se refiere a datos diferentes cada vez que se emplea modificando el valor básico (TABLA) mediante el valor indexado (I), que va cambiando reiteradamente.

El principio fundamental del *direccionamiento indexado* es que el contenido del registro índice se agrega a la dirección de base proporcionada junto con la instrucción para obtener así la *dirección efectiva*, es decir, la posición de memoria a la que se accederá en ese instante. Si esta instrucción se produce dentro de un bucle, entonces se puede realizar la modificación por medio del registro índice (incrementos y decrementos por lo general) incluso dentro del mismo bucle, por lo que es fácil acceder a toda una tabla de valores.

El 6809 no sólo tiene dos registros para este menester, el X y el Y, sino otros dos más, el S y el U. Incluso en circunstancias especiales se puede echar mano del contador del programa. Pero el tema del direccionamiento indexado se hace todavía más enredado si se piensa que hay varios tipos de indexación. Con ellos se cubren casi todas las necesidades de programación. De aquí en adelante utilizaremos la indexación de una u otra forma, por lo que tendremos magníficas oportunidades para familiarizarnos con los diferentes modos en que se utiliza.

El direccionamiento indexado se indica colocando X en el campo para operandos, si es que nos servimos del registro X como índice. La forma general de una instrucción indexada sería la siguiente:

```
Opcode   Desplazamiento, Registro índice
LDA      TABLA1,X
STA      TABLA2,Y
```

En muchas ocasiones el desplazamiento es cero, en cuyo caso puede omitirse. Por ejemplo:

```
Opcode   ,Registro índice
LDA      ,X
STA      ,Y
```

Veamos cómo funciona en la práctica. Supongamos que tenemos una tabla de 64 valores de ocho bits almacenados a partir de la posición $3000 y deseamos acceder secuencialmente a los bytes. Podemos definir la dirección base y reservar espacio para la tabla mediante directivas:

```
        ORG     $3000
TABLA   RMB     64
```

Estas instrucciones ponen el contador del programa a $3000, definen el inicio de la tabla en $3000 (TABLA) y reservan los 64 bytes siguientes. Accederemos ahora a los bytes mediante el siguiente paso: el nuevo ORG significa que el código subsiguiente será almacenado en otra parte distinta de la memoria. Cuando se comienza a usar el direccionamiento indexado resulta una medida de precaución recomendable contra la pérdida de control de un bucle, impidiendo así que el programa se destruya a sí mismo:

```
        ORG     $1000
CONT    FCB     0
        LDX     #0
BUCLE   LDA     TABLA,X
```

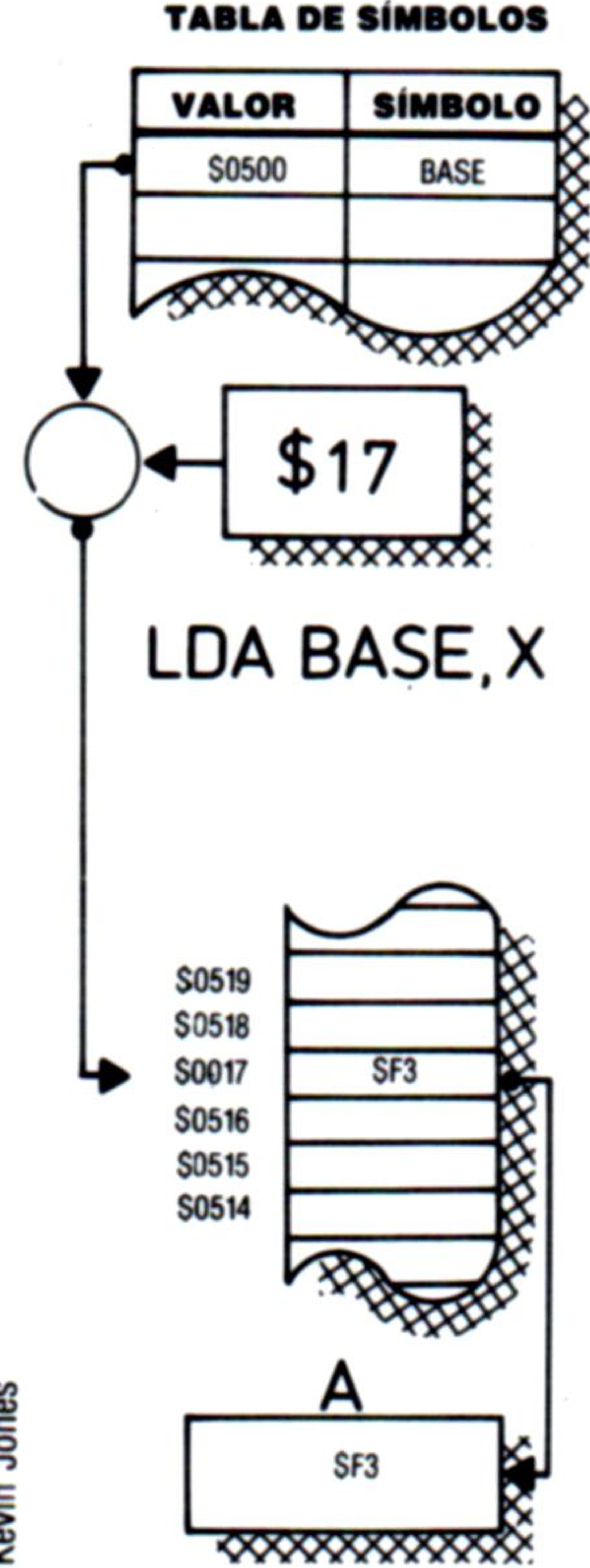
Alteraremos ahora el valor del registro X:

```
        TFR     X,D
        ADDD    #1
        TFR     D,X
```

Sin duda es ésta una manera algo laboriosa de incrementar X, aunque se revela útil cuando se trata de incrementar o decrementar con números mayores de dos. Más adelante estudiaremos otras alternativas a este método. El último fragmento del programa incrementará el contador y hará una comprobación por si se llegó ya al 64 (en cuyo caso se concluye el programa y no hay necesidad de ejecutar el bucle):

```
        INC     CONT
        LDB     CONT
        CMPB    #64
        BLT     BUCLE
```

Enlaces índices
El símbolo BASE, que se inicializa con $0500, es la dirección del primer byte en una tabla de valores. La instrucción en modo de direccionamiento indexado LDA BASE,X toma el valor de BASE y le agrega el contenido del registro X, para obtener la dirección específica del byte cuyo contenido se carga en el acumulador. Si esta instrucción se coloca dentro de un bucle del que X hace de contador, entonces es posible acceder a la tabla entera, byte a byte secuencialmente. Dado que X es un registro de 16 bits, el bucle puede cubrir todo el espacio de la memoria (desde $0000 hasta $FFFF en un sistema de ocho bits, como es el caso del 6809)



Kevin Jones



Existen varias maneras de mejorar la eficacia de este código. Entre las más útiles se encuentra el modo de *autoincremento*. La instrucción

```
        LDA        TABLA,X+
```

hará que el valor contenido en X se incremente automáticamente después de ser utilizado. Si tenemos una tabla de valores de 16 bits tendremos que escribirla así:

```
        LDA        TABLA,X++
```

con lo que el registro X se incrementa en dos. Nuestro programa original resulta ahora mucho más fluido:

```
BUCLE   LDA        TABLA,X+
        INC        CONT
        LDB        CONT
        CMPB       #64
        BLT        BUCLE
```

Una alternativa también útil al método explicado al principio sería recorrer los valores de la tabla en orden inverso, utilizando el modo de autodecremento. Tiene la ventaja de que el valor final del registro X es cero, y dado que el autodecremento de un registro índice activa los flags del registro de código de condición automáticamente, nos será fácil detectar el final del bucle sin necesidad de usar la instrucción CMP. Esto mismo se puede obtener cargando el registro índice con un valor negativo e incrementándolo hasta llegar a cero. Cada vez que se cumple la instrucción de autoincremento, se activan los flags del registro de código de condición para mostrar el resultado del incremento. Si éste es cero, por ejemplo, el que se activa es el flag cero; si se produce un arrastre *(carry)*, se activa el flag de arrastre, etc.

Pero no debemos olvidar la regla general de que es mejor hacer sólo verificaciones de los acumuladores. Y esto porque muchos programas pueden intercalar algún proceso entre la instrucción de incremento (decremento) y la instrucción verificadora, lo cual hace improbable que el registro de código de condición no sufra modificaciones en el código que va de la instrucción a la verificación.

Si decidimos no recorrer la tabla al revés, todavía lo que es posible hacer es que el contador vaya bajando hasta acabar el bucle en cero. Pero hay que estar atentos a que en el modo de autodecremento este último se realiza *antes* del cálculo de la dirección, mientras que en el autoincremento el registro se incrementa *después* de haber sido calculada la dirección. Así, suponiendo que X contiene un 7 y la TABLA comienza en $1000, la instrucción LDA TABLA,X+ cargará el acumulador con el valor contenido en $1000 y seguidamente incrementará X, que pasará del 7 al 8. Por otra parte, LDA TABLA,–X (el signo menos delante del nombre del registro), primero cambiará X del 7 al 8 y después cargará el acumulador con el valor contenido en la dirección $1006.

Nuestro bucle tendrá el siguiente aspecto, si recorremos la tabla al revés y guardamos el contador en el registro B:

```
        LDX        #64
        LDB        #64
BUCLE   LDA        TABLA,-X
        DECB
        BGE        BUCLE
```

El primero de los dos programas que hemos ofrecido como ejemplo muestra un bucle directo que recorre una tabla de valores de ocho bits, donde se cuenta el número de valores negativos. La cuenta llevada por el acumulador B es también empleada como el desplazamiento de un valor fijado en el registro X.

El segundo programa muestra el uso de ambos registros índice conjuntamente, con un desplazamiento cero. Copia la cadena de caracteres colocada en una posición (puede ser en un buffer de entrada, por ejemplo) a otra posición donde será almacenada. La longitud de la cadena es desconocida (contando siempre con que sea menor de 255 bytes), pero concluirá con un carácter RETURN. Una vez almacenado en sus posiciones finales, este carácter RETURN será borrado y se agregará al inicio de la cadena un byte que indique la longitud hasta ese momento desconocida.

Contador de valores negativos

• Se almacena en $3001 una tabla de valores de ocho bits. En $3000 se almacena el número de valores que contiene la tabla que se supone menor de 256. Este programa cuenta el número de valores negativos que contiene la tabla

```
TABLA   EQU   $3000     TABLA contiene el numero de valores
        ORG   $1000
NEGS    FCB   0         NEGS es el lugar donde se guarda el contador de los valores negativos hallados
        LDB   TABLA
        LDX   #TABLA    La dirección de la tabla va a X
BUCLE   LDA   B,X       Toma de la tabla el ultimo valor
        BGE   FINLP     Si el valor es positivo GOTO FINLP
        INC   NEGS      Si no, se cuenta como valor negativo
FINLP   DECB            Recorrido de la tabla
        BGT   BUCLE     Hay mas valores? Si asi es, GOTO BUCLE
        SWI             Si no, se vuelve al sistema operativo
        END
```

Copiador de cadenas de caracteres

• Este programa copia una cadena desde un buffer de entrada en $3000 al siguiente espacio libre para cadenas, cuya dirección se da en $0010

```
CR      EQU   13         Valor ASCII del caracter RETURN
STRING1 EQU   $3000      Dirección del buffer de entrada
STRING2 EQU   $0010      STRING2 retiene la dirección del espacio libre
        ORG   $1000
        LDX   #STRING1   Direccion de la cadena fuente (o sea, $3000 al principio) cargada en X
        LDY   STRING2    Direccion de la cadena de destino (almacenada en $10,$11) cargada en Y
        TFR   Y,U        Paso a U de la direccion del byte de longitud
        LDA   #0         Almacena un cero en el primer
        STA   ,Y+        byte de la cadena de destino
BUCLE   LDA   ,X+        Toma el caracter siguiente del buffer de entrada
        CMPA  #CR        Caracter RETURN?
        BEQ   FINAL      Si no es asi, parar
        STA   ,Y+        Si no es asi, copiarlo (y)
        INC   ,U         Añadir uno a la longitud
        BRA   BUCLE      Caracter siguiente
FINAL   SWI              Volver al sistema operativo
END
```

Apostando al favorito

Este programa de carreras de caballos, creado por Salamander Software para el Oric-1 y el Atmos, es un claro ganador









Ganando etapas
En cualquier etapa del juego los jugadores pueden consultar el programa de carreras (foto superior). Éste proporciona detalles sobre el desarrollo de la prueba y también indica el montante de los premios para carreras específicas. Después de que se han escogido los corredores para una carrera determinada, los jugadores tienen la oportunidad de apostar al caballo de su elección (segunda fotografía). Las apuestas deben oscilar entre 5 y 500 libras. Los caballos se alinean entonces en espera del disparo de salida, y los jugadores se reclinan en sus asientos y aguardan a que empiece la carrera. En la fotografía final vemos a los caballos frenándose tras superar el poste de llegada, con el elegido de nuestro fotógrafo situado en una triste cuarta posición...

Classic racing (Hípica clásica) le ofrece la oportunidad de jugar el papel de un entrenador de caballos de carreras durante una temporada de encuentros. Es un juego para 1-6 jugadores, pero, cuando hay menos de seis personas jugando, el ordenador completa los números, de modo que cada carrera cuenta con seis participantes. El juego le permite elegir la duración de la temporada: una temporada completa abarca 16 reuniones hípicas, cada una de ellas de seis carreras. Quienes tengan menos paciencia seleccionarán una temporada más corta. El objetivo del jugador consiste en ganar la mayor cantidad de dinero posible. Esto se puede conseguir de dos formas: obteniendo el dinero del premio entrenando a uno de los tres primeros caballos de una carrera, o bien haciendo apuestas sobre el resultado. Cada jugador *debe* entrar un corredor para cada carrera, pero no hay nada que le impida apostar a alguno de los corredores de sus contrincantes en caso de que piense que ello le significará una mejor oportunidad de ganar dinero.

Su cuadra consta de 16 caballos y al comienzo de la temporada no se tiene ningún conocimiento acerca de sus méritos. Dado que en las primeras carreras de la temporada los premios son pequeños, ésta es la época ideal para experimentar probando a sus corredores en distintas distancias y con diferentes características de suelo. Esto es simplemente cuestión de ensayo y error: debe observar el rendimiento de un caballo determinado en ciertas condiciones y planificar su estrategia en consecuencia. Ello supone tomar una gran profusión de notas; cada vez que corre un caballo es conveniente apuntar la distancia, el peso transportado, el estado del suelo y el resultado. Es una lástima que Salamander no haya logrado incluir una subrutina para imprimir tales detalles de forma automática, porque ello supondría el ahorro de muchísimo esfuerzo.

Una vez el jugador ha elegido sus seis corredores para la primera reunión hípica, se le darán los nombres de los caballos de sus oponentes y el peso que transportará cada uno. El ordenador hace entonces las apuestas contra cada caballo ganador. Al principio de la temporada esto parecerá hacerse de un modo arbitrario, pero a medida que avanza la temporada, los caballos con malos tiempos partirán con apuestas más bajas. Las apuestas son obligatorias y deben estar entre 5 y 500 libras esterlinas, y las probabilidades ofrecidas pueden ser muy generosas. Dado que un caballo ganador (o colocado) ganará dinero en premio, es ventajoso apostar a alguno de los caballos de los oponentes, con lo cual se tienen dos posibilidades de obtener un beneficio.

También se pueden fraguar "estratagemas" de apuesta, introduciendo a un caballo en carreras para las que obviamente no es adecuado; por ejemplo, un caballo que se desempeñe bien en mil metros en terreno pesado se podría entrar para dos carreras sucesivas de 2 500 m en terreno firme. Es casi seguro que perdería ignominiosamente, después de lo cual se lo podría introducir en una carrera más apropiada con buenas posibilidades. Sin embargo, una vez que se han decidido la distancia y el terreno ideales para un caballo determinado, debe resistirse la tentación de hacerlo correr una carrera tras otra: al igual que en el mundo real de la hípica, los caballos necesitan descansar con frecuencia para obtener su mejor rendimiento.

Cuando se han hecho todas las apuestas, la acción pasa a la carrera propiamente dicha. Los caballos se dirigen a paso cómodo hasta sus posiciones, el juez de salida los llama al orden y después las facilidades de sonido del Oric producen una aproximación bastante fidedigna a los ruidos de los cascos sobre la pista cuando los corredores avanzan hacia la meta. La secuencia de la carrera está lograda de una forma maravillosa: los caballos forcejean por sus posiciones de forma realista y los corredores son tan proclives a tener una reacción impredecible como sus equivalentes de la vida real.

Al final de la carrera se abonan las apuestas ganadoras y el proceso se repite para el resto del programa hasta que finaliza la reunión hípica. Cada encuentro tiene una pista con distintas condiciones de suelo y distancias. Si finalmente decide que uno de sus caballos no está en forma, lo puede retirar de su nómina simplemente no haciéndolo correr en tres encuentros sucesivos. Esto le supone un factor menos de preocupación, pero le vale una penalización de mil libras por cada uno de los encuentros que aún falten.

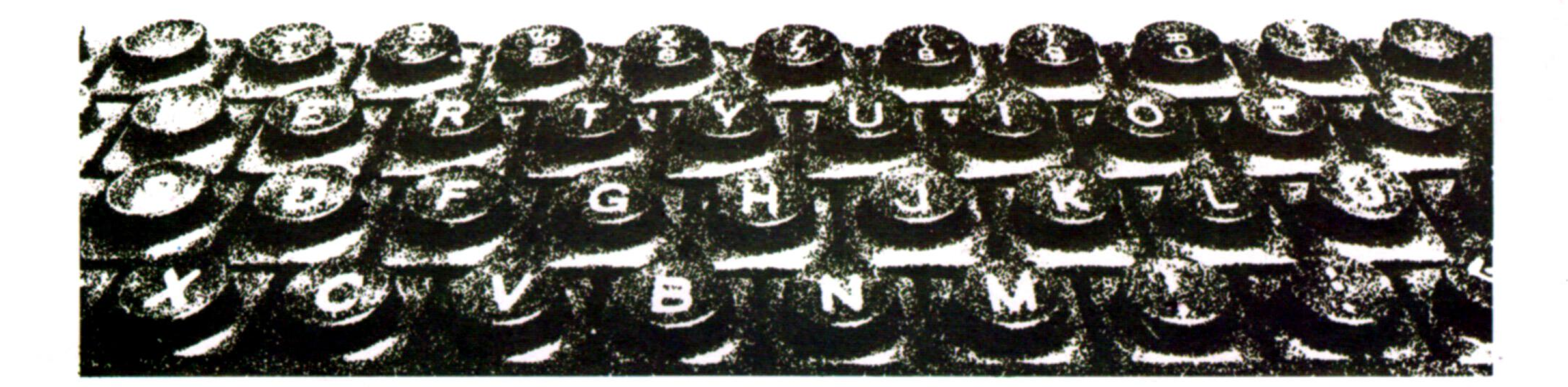
Hacia el final de la temporada, las carreras van resultando más difíciles de ganar, puesto que para entonces todos los jugadores tienen un concepto mucho más claro acerca del rendimiento de sus caballos y es menos probable que introduzcan corredores en carreras en las que no tienen posibilidades de ganar. Las recompensas son consiguientemente mayores: los tres primeros caballos del Derby, que se corre durante el último encuentro de la temporada, se reparten 90 000 libras en premios.

Classic racing es el programa más atractivo que existe hasta el momento para el Oric y el Atmos. Las secuencias de las carreras son de visión obligatoria, y la planificación total del juego posee un *crescendo* tan apasionante, que hace que el juego siga despertando gran interés aun después de haberlo jugado varias veces.

Classic racing: Para el Oric-1/Atmos de 48 K
Editado por: Salamander Software, 17 Norfolk Road, Brighton BN1 3AA, Gran Bretaña
Autor: Paul Neal
Palancas de mando: No se necesitan
Formato: Cassette

mi computer

Editorial Delta, S.A.



10054
9 788485 822836